COMPUTE-IN-MEMORY WITH EMERGING NON-VOLATILE MEMORIES FOR ACCELERATING DEEP NEURAL NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Xiaoyu Sun

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2020

COMPUTE-IN-MEMORY WITH EMERGING NON-VOLATILE MEMORIES FOR ACCELERATING DEEP NEURAL NETWORKS

Thesis committee:

Dr. Shimeng Yu, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Arijit Raychowdhury
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jae-sun Seo
School of Electrical, Computer and Energy Engineering
Arizona State University

Dr. Shaolan Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: July 6, 2020

ACKNOWLEDGMENTS

Firstly, I would like to especially thank my advisor Dr. Shimeng Yu for the continuous support and guidance throughout this whole journey. He has been giving me a full-range of guidance from defining research objectives, forming research ideas, to paper writing and oral presentation skills. Besides the help on research, he is always a nice and thoughtful advisor who cares about students. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my dissertation committee: Dr. Sung Kyu Lim, Dr. Shaolan Li, Dr. Arijit Raychowdhury and Dr. Jae-sun Seo, for their insightful and valuable comments.

I would like to thank my former and current labmates at both Arizona State University and Georgia Tech for the wonderful time that we experienced together: Dr. Ligang Gao, Dr. Pai-Yu Chen, Dr. Rui Liu, Dr. Jiyong Woo, Dr. Zhiwei Li, Xiaochen Peng, Panni Wang, Shanshi Huang, Hongwu Jiang, Bin Dong, Dr. Wonbo Shim, Dr. Jae Hur, Yandong Luo, Wantong Li, Anni Lu, Yuan-Chun Luo, and Gihun Choe. I would also like to extend my thanks to Shihui Yin, Xiaocong Du, Dr. Manqing Mao, Dr. Win-San Khwa, Gauthaman Murali, Muya Zhang, and Dr. Jong-hyeok Yoon for the exciting collaborations that we did together.

I'm also grateful to Dr. Kerem Akarvardar and TSMC for giving me the opportunity to intern at the leading company in the industry.

Last but not the least, I would like to thank my parents and all the family members for supporting and encouraging me throughout this whole journey. I would not have made it without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this research is to accelerate deep neural networks (DNNs) with emerging non-volatile memories (eNVMs) based compute-in-memory (CIM) architecture. The research first focuses on the inference acceleration and proposes a resistive random access memory (RRAM) based CIM architecture. Two generations of RRAM testchips which monolithically integrate the RRAM memory array and CMOS peripheral circuits are designed and fabricated using Winbond 90 nm and TSMC 40 nm commercial embedded RRAM process respectively. The first generation of testchip named XNOR-RRAM is dedicated for binary neural networks (BNNs) and the second generation named Flex-RRAM features 1bit-to-8bit run-time configurable precision and leverages the input sparsity of the DNN model to improve the throughput and energy-efficiency. However, the non-ideal characteristics of eNVM devices, especially when utilized as multi-level analog synaptic weights, may incur a notable accuracy degradation for both training and inference. This research develops a PyTorch based framework that incorporates the device characteristics into the DNN model to evaluate the impact of the eNVM nonidealities on training/inference accuracy. The results suggest that it is challenging to directly use eNVMs for in-situ training and resistance drift remains as a critical challenge to maintain a high inference accuracy. Furthermore, to overcome the challenges posed by the asymmetric conductance tuning behavior of typical eNVMs, which is found to be the most critical nonideality that prevents the model from achieving software-equivalent training accuracy, this research proposes a novel 2-transistor-1-FeFET (ferroelectric field effect transistor) based synaptic weight cell that exploits hybrid precision for in-situ training and inference, which achieves near-software classification accuracy on MNIST and CIFAR-10 dataset.

CHAPTER 1
INTRODUCTION

Deep neural networks (DNNs) have achieved unprecedented success in various intelligent tasks including computer vision [1, 2, 3], speech recognition [4], natural language processing [5, 6], and autonomous driving [7] etc. DNNs are known for involving a substantial amount of parameters and highly intensive computations and they are getting deeper and deeper rapidly to gain incremental performance improvement. For example, ResNet [3] could contain more than 150 layers and over 60M parameters. This exponential growth of model size poses grand challenges on efficient hardware implementations due to the intensive computations and massive data communications between the processor elements and memory, which is known as the memory-wall bottleneck [8] in traditional von Neumann architecture.

The innovations in computing hardware have been conducted to tackle the challenges. Graphics processing units (GPUs) are extensively used in deep learning tasks [9], which enhance the parallelism of computation. Also, the memory chips with enhanced density and bandwidth such as the hybrid memory cube (HMC) [10], and high bandwidth memory (HBM) [11] are deployed to mitigate the memory bottleneck. Meanwhile, application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) based accelerators have been designed to match the deep learning algorithms and dataflow [12, 13, 14, 15, 16, 17]. For example, Google released the ASIC processor TPU [12] for performing practical workloads in datacenters that outperforms the GPUs based on their benchmark. While showing better performance compared to traditional general purpose processors, the energy-efficiency of these designs is still ultimately limited by the data communications between the separated memory and logic. Thus, to really break the memory-wall, it's of great interests to embed the computation into the memory itself [18], namely compute-

in-memory (CIM), to minimize the data transfer or even eliminate the off-chip memory access. In CIM, the multiply-and-accumulate operation, which is known as the dominant computation in DNNs, is performed by asserting multiple or all the rows of the memory array simultaneously and the weighted-sum is added up through the column current in analog domain. Analog-to-digital converters (ADCs) are typically deployed at the edge of the memory array to convert the analog current/voltage to digital binary code for further processing (e.g., shift-and-add, activation, and pooling).

For CIM implementations, static random access memory (SRAM) based designs (typically include several additional transistors) have been proposed [19, 20, 21, 22, 23, 24, 25], which demonstrate much higher energy-efficiency compared to the mainstream GPUs. However, one SRAM cell typically consumes more than $150F^2$ (F is the feature size of a technology node), making the on-chip memory capacity too limited to hold all the weights of state-of-the-art DNNs. In addition, SRAM is volatile and suffers from increasing leakage power in scaled CMOS nodes. To this end, emerging non-volatile memories (eNVMs), including resistive random access memory (RRAM), spin-transfer-torque magnetic random access memory (STT-MRAM), phase change memory (PCM), and ferroelectric field effect transistor (FeFET), are more suitable candidates due to the non-volatility and high density. Moreover, the multi-level programmability of eNVM devices can further improve the area-efficiency. Industry has been continuously investing in eNVM technologies and several commercial processes have been announced such as Intel's 22nm RRAM [26], TSMC's 40nm RRAM [27], Intel's 22nm STT-MRAM [28], Samsung's 28nm STT-MRAM [29], and TSMC's 40nm PCM [30]. Meanwhile, doped-HfO_2 based FeFET is also rapidly emerging, e.g., GlobalFoundries' 22nm FeFET [31]. Thanks to these progresses, several eNVM based CIM chip-level demonstrations have been successfully delivered in recent years [32, 33, 34, 35]. In this research, we design two generations of RRAM based CIM testchips, namely XNOR-RRAM chip [36] and Flex-RRAM (in fabrication), as the pioneering works in the field. The testchips are taped-out with Winbond's 90 nm RRAM

process [37] and TSMC's 40 nm RRAM process [30] respectively. XNOR-RRAM chip is dedicated for binary neural networks (BNNs) and Flex-RRAM chip features 1bit-to-8bit configurable precision at run-time. The measurement and simulation results show promising improvements in terms of throughput and energy-efficiency.

However, it should be noted that the eNVM devices exploited in those testchips are typically in binary state for reliability concern, which constrains the efficiency of the design. Also, the measured accuracy result typically does not take reliability issues into consideration such as temperature and retention etc. While there are device-level demonstrations of multi-level capability [38, 39, 40], analog eNVM devices typically suffer from reliability and uniformity issues, which may degrade the accuracy performance when utilized as synaptic weights. For example, PCM is known for strong temperature dependence [41] and severe resistance drift [42]. Moreover, it becomes more challenging when the eNVM based weights need to be trained on the fly, so-called in-situ training. The non-idealities including device-to-device (D2D) variation, cycle-to-cycle (C2C) variation, nonlinear and asymmetric conductance tuning, and write endurance will lead to the deviation of weights from the desired value, thus degrading the training accuracy. In this research, we conduct a comprehensive investigation on the impact of eNVM device non-idealities on training and inference with a in-house PyTorch-based simulation framework, where the device non-idealities are modeled by generalized numerical models and incorporated into the DNN models [43]. The results suggest that non-ideal device characteristics are critical roadblockers to achieving software-equivalent accuracy for both in-situ training and inference.

Recent works have tried to address the challenge posed by device non-idealities from device, circuit and algorithm perspective. From device perspective, Wu et al. [44] presented a methodology that improves the linearity of analog RRAM conductance tuning by inserting an additional electro-thermal modulation layer into the material stack. In [45], optimized pulse schemes with non-identical pulse width or pulse amplitude were proposed to improve the linearity and symmetry of the conductance tuning of FeFET.

At circuit level, capacitor-assisted designs (e.g., 3-transistor-1-capacitor (3T1C) [46] and 3T1C+2PCM [47]) were proposed to maintain the software-equivalent training accuracy. At algorithm level, Huang et al. [48] showed that by exploiting the adaptive momentum in the weight update rule, the asymmetry of the conductance tuning becomes less destructive. In this research, we propose a solution through software-hardware co-design. A synaptic weight cell that combines two CMOS transistors and one FeFET (2T1F) is designed and the training and inference are performed at hybrid precision [49]. During training, the "volatile" modulated gate voltage of FeFET is used to represent LSBs for symmetric and linear update. After training process is done, the information of LSBs is discarded, only MSBs are preserved by "non-volatile" polarization states of FeFET for inference. The simulation result shows that the training accuracy could achieve $\sim$97.3%/$\sim$87% on MNIST [50] and CIFAR-10 [51] dataset respectively, approaching the ideal software training.

The rest of the thesis is organized as follows: Chapter 2 introduces the related background and prior works. Chapter 3 presents two generations of RRAM based CIM testchips for inference application. Chapter 4 presents the investigation on the impact of device non-idealities on training and inference accuracy. Chapter 5 presents the 2T1F weight cell design for training and inference at hybrid precision. Finally, Chapter 6 concludes the thesis with key contributions and future directions.

CHAPTER 2

BACKGROUND

2.1 Brief Introduction of Neural Networks

An Artificial neural network (ANN) is a computational model inspired by the biological neural networks in the human brain [52]. Thanks to the unprecedented breakthroughs various tasks such as computer vision and natural language processing, ANNs have attracted tremendous attentions from both academia and industry in the past 10 years. The most widely used ANN models include multi-layer perceptron (MLP) [53], convolutional neural network (CNN) [54], and recurrent neural network (RNN) [55]. In this section, we briefly introduce the structure and working principle of neural networks.

2.1.1 A Single Neuron

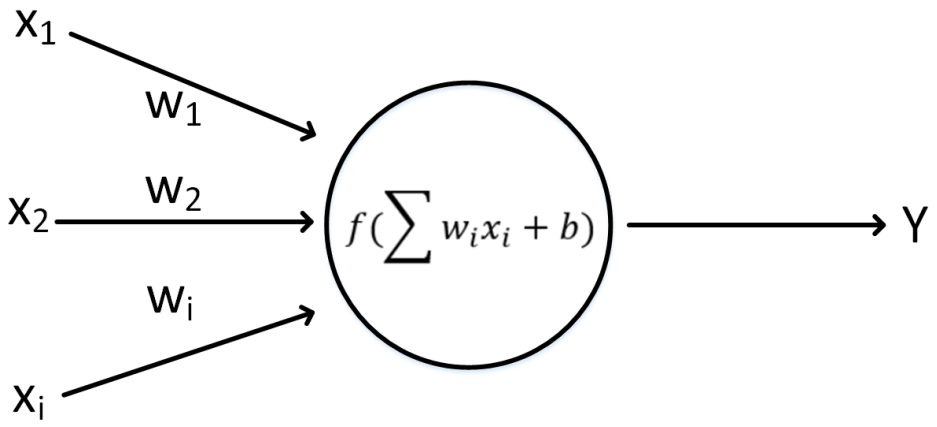


Figure 2.1: Diagram of a single neuron in neural networks.

The most basic unit of an ANN is a neuron. ANNs typically consist of a huge number of neurons and they are connected with synapses (also known as weights). Figure 2.1 depicts the principle of a single neuron, which receives input x_i from the connected neurons in the

previous layer and each connection has an associated weight w_i. The neuron then applies a function f, known as the activation function, to the weighted-sum of inputs and weights (with an optional bias b) . The most widely used activation function include sigmoid, tanh, and rectified linear unit (ReLU). Then the output Y is generated and sent to the neurons in the next layer.

2.1.2 Multi-layer Perceptrons

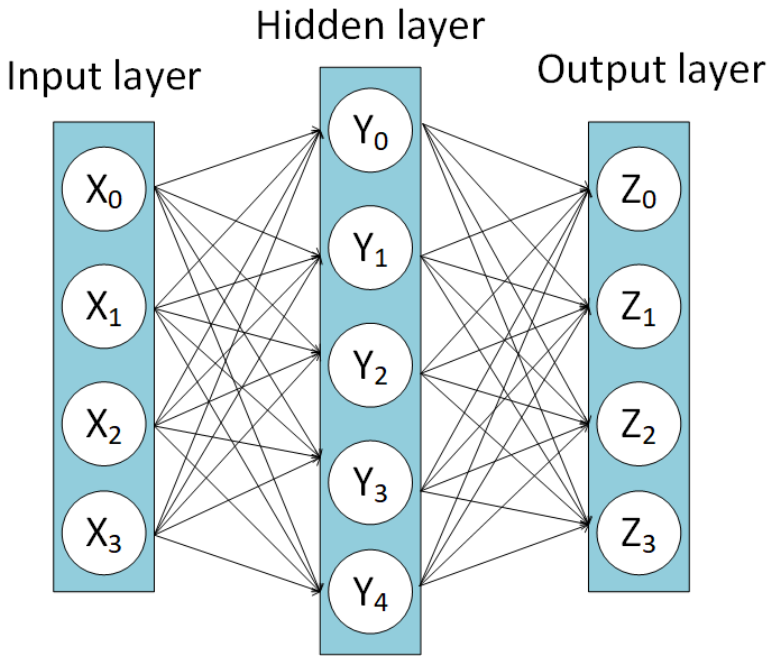


Figure 2.2: Diagram of a multi-layer perceptron with one hidden layer.

A multi-layer perceptron contains neurons arranged in layers and the neurons in adjacent layers are connected through synapses, where every connection has an associated weight. Figure 2.2 shows the structure of a MLP with one hidden layer as an example. The input layer provides the external input information to the network, e.g., the pixel values when the input is a image. The output layer delivers the final output for evaluation, e.g., the probability of each class when used for image classification. The hidden layer is the layer in between, which can be of one or more layers. A MLP is also called a fully connected neural network when the neurons in adjacent layers are fully connected, i.e., each neuron

is connected to all the neurons in the adjacent layer. However, pruning techniques [56, 57] are typically utilized in practice to reduce the number of parameters in the model to save area and energy.

2.1.3 Convolutional Neural Networks

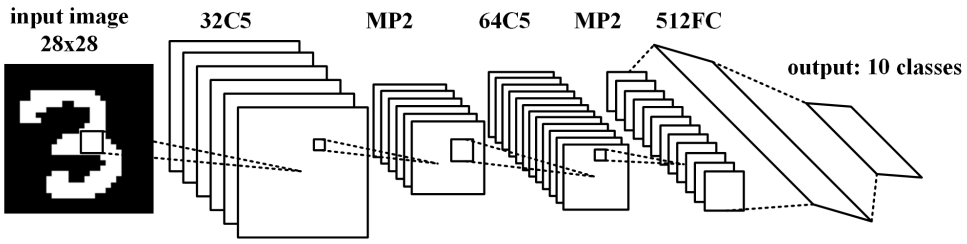


Figure 2.3: Diagram of a convolutional neural network for MNIST handwritten digit classification. The network consists of two convolutional layers, two fully connected layers and two pooling layers.

Nowadays, CNNs are the most widely used neural network topology due to the excellent feature extraction capability, especially in computer vision applications. For example, AlexNet [1], VGG-Net [2], GoogleNet [58], and ResNet [3] are well-known CNN models that won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [59] in the past years.

Figure 2.3 shows the diagram of a CNN, which is a variant of LeNet-5 [60], as an example. It consists of multiple convolutional (CONV) layers to extract the features of the input data, followed by a relatively small number (e.g., 1 to 3) of fully-connected (FC) layers as classifiers. The main computation of CNN is the convolution operation between a set of filters and input feature maps (IFMs), which generates the output feature maps (OFMs). A pooling layer is utilized to down-sample the OFMs to gradually reduce the dimension size of the feature representation.

2.1.4 Training and Inference

Training and inference are two phases of the deployment of a neural network. Training refers to the process of updating the values of weights and biases through feedforward and backpropagation, which typically involves a training dataset with inputs and output labels, a loss function and a training algorithm. For example, stochastic gradient descent [61] is the most successful training algorithm that has been widely used in the training of neural networks, through which the weights and biases are updated to minimize the error calculated by the loss function. The training phase typically takes a large number of epoches for convergence, thus is a highly computation-intensive process that is always performed in servers or datacenters. All the trainable parameters are fixed once the training is done.

Inference refers to the phase of using the trained neural network model to process the input data from practical use-case, which is never exposed to the model, to make predictions or decisions. Comparatively, inference incurs much less computations as only feedforward computation is involved, thus allowing the execution on edge devices such as smart phones and wearable devices.

2.2 Emerging Non-volatile Memory for Synaptic Devices

eNVM devices have been proposed to implement synaptic weights on-chip for the higher density (typically 4–12 F^2 bit cell) and the feasibility of highly parallel analog computing with low leakage power consumption [18]. The candidates include the two-terminal eNVMs such as PCM, RRAM, and the three-terminal FeFET. These eNVMs have been actively explored as promising candidates for possible replacement of NAND/NOR FLASH in memory storage applications. Although not applied to massive volume products yet, some prototype chips have been demonstrated. For example, Samsung reported an 8-Gb PCM prototype chip at 20 nm with a write bandwidth of 40 MB/s [62], Intel reported a 10.1

Mb/mm^2 RRAM macor at 22 nm FinFET technology [26]. However, the requirement for eNVM devices to implement synaptic weights is more stringent than using as binary memory device as multi-level conductance states is desired to efficiently represent the weight, especially for in-situ training acceleration. In this section, we survey the aforementioned eNVM candidates for implementing synaptic weights with a focus on the analog multi-level programmability.

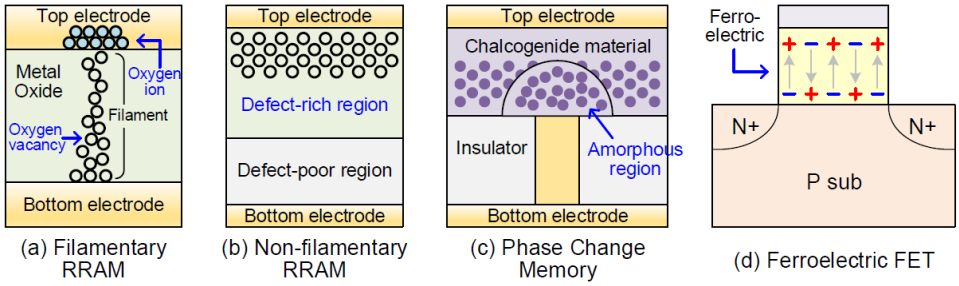


Figure 2.4: Structure diagram of (a) filamentary RRAM, (b) interfacial RRAM, (c) PCM, and (d) FeFET. Adapted from [63]

2.2.1 RRAM

RRAM is a non-volatile two-terminal device technology where the cell resistance can transition between the high resistance state (HRS) and the low resistance state (LRS). Generally, the resistance is increased and decreased by applying voltage pulses with different polarities, which is referred to as RESET and SET process respectively. There are typically two main categories of switching mechanisms of RRAM devices: one is the so-called filamentary mechanism which relies on the formation and rupture of the conductive filament that consists of metal ions or oxygen vacancies [64, 65, 66, 67, 68] as shown in Figure 2.4(a); the other one is based on the interfacial mechanism where the distribution of oxygen vacancies at the interface is modulated by the electric field [69, 70, 71, 72] as shown in Figure 2.4(b). Filamentary RRAM is widely adopted for digital memory application for its fast and low-power switching process. However, the formation of the filament, i.e., the SET process, is typically abrupt due to the positive feedback between the growth of

the filament and the electric field. Thus, a unidirectional RESET-only updating scheme was adopted in the HfO_x-based synaptic device [73]. In order to make the SET process gradual, bilayer-oxide structures (e.g., TaO_x/HfO_2 [74] and AlO_x/HfO_2 [75]) have been proposed to form multiple weak filaments instead of one single strong filament. On the other hand, the interficial RRAM shows gradual SET and RESET process. However, the resistance tuning curve typically suffers from nonlinearity and asymmetry, thus requiring more complex programming schemes with non-identical pulses for mitigation.

2.2.2 PCM

The resistance change in PCM relies on the chalcogenide materials, typically $Ge_2Sb_2Te_5$, to switch between the amorphous phase (high resistance state) and the crystalline phase (low resistance state). Its device structure is shown in Figure 2.4(c). By controlling the volume of the amorphous region, the multi-level resistance states could be achieved, thus enabling the implementation of analog weights. One critical challenge of the PCM-based synaptic device is the relatively more abrupt RESET process compared to the gradual SET process as the melting and quench process happened during RESET is less controllable than the partial crystallization process happened during SET. To address this challenge, one approach from architecture perspective is to use two PCM devices as one weight [76, 77], where the gradual SET process is utilized in both devices and the weight increase and decrease are realized through differential read-out. On the other hand, using non-identical pulse schemes can enhance the multi-level programmability in both directions as demonstrated in [78].

2.2.3 FeFET

FeFET is a three-terminal device, which resembles a MOSTFET in structure, except a additional ferroelectric oxide layer is deposited in the gate-stack as shown in Figure 2.4(d). Thanks to the much improved CMOS compatibility and scalability, the HfO_2 thin film

based FeFET has been under active research in recent years for potential applications in both storage and computing.The polarization state of the Si:HfO_2 thin film could be flipped by applying positive or negative electrical field across the thin film. Thus the gate capacitance and consequently the threshold voltage of the FeFET can be gradually tuned by applying corresponding voltage pulses to the gate. To program the cell, a positive gate voltage should be applied while grounding the substrate. Conversely, a negative gate voltage should be applied to erase the cell. Nevertheless,to avoid the on-chip generation of negative voltage, grounding the gate and apply positive voltage to the body would do the job as well. The work [45] experimentally demonstrate the tunable channel conductance with a $Hf_{0.5}Zr_{0.5}O_2$ (HZO) based FeFET. Several different pulse schemes are explored, including identical pulses, pulses with increasing pulse-width, and pulses with increasing amplitude. The results suggest that the non-identical pulse schemes can improve the linearity and symmetry of the tuning curve. In addition, as compared with RRAM, FeFET shows several better features such as larger ON/OFF ratio, shorter programming pulse-width, and less programming energy consumption.

2.3 Chip-level Demonstrations of eNVM-based CIM Accelerator

In recent years, extensive works have been done in the field of eNVM-based CIM for DNN acceleration, including device engineering and modeling, circuit-level innovation, architectural optimization, chip-level demonstration, and benchmarking framework development. In this section, we survey the recent progresses on macro-/chip-level demonstrations that have shown great promises on practical deployment.

Table 2.1 summarizes the representative macro-/chip-level demonstrations of the eNVM-based CIM in the past few years. In 2015, Prezioso et al. [79] demonstrated a 12×12 RRAM crossbar array that implemented a single-layer perceptron with ten inputs and three outputs. The network was tested on classifying 3 groups of stylized letters (z, v, and n), each group consists of the correct pattern and nine additional noisy patterns formed by

Table 2.1: Survey of Recent Chip-level Demonstrations of eNVM-based CIM

Affiliation	Year	eNVM	Tech node	Array size	Cell precision	TOPS/W	Dataset	Accuracy	Training/inference
UCSB [79]	2015	RRAM	N/A	12×12	3-bit	N/A	3×3 B/W image	N/A	Training
IBM [76]	2015	PCM	$180nm$	500×661	4-bit	N/A	MNIST	82.9%	Training
Tsinghua [80]	2017	RRAM	$1.2\mu m$	128×8	3-bit	N/A	Yale Face	91.5%	Training
Tsinghua/ASU [81]	2017	RRAM	$130nm$	512×1024	1-bit	N/A	MNIST	96.5%	Inference
UMass [82]	2018	RRAM	$2\mu m$	128×64	7-bit	N/A	MNIST	91.7%	Training
Panasonic [83]	2018	RRAM	$40nm$	$4Mb$	3-bit	66.5	MNIST	90.8%	Inference
NTHU [32]	2018	RRAM	$65nm$	512×256	1-bit	N/A	MNIST	96.2%	Inference
Umich [84]	2019	RRAM	$180nm$	54×108	7-bit	0.2	N/A	N/A	Training
NTHU [33]	2019	RRAM	$55nm$	256×512	1-bit	53.2	CIFAR-10	81.8%	Inference
ASU/Gatech [36]	2019	RRAM	$90nm$	128×64	1-bit	24.1	CIFAR-10	83.5%	Inference
Tsinghua [35]	2020	RRAM	$130nm$	$158.8kb$	1-bit	78.4	MNIST	94.4%	Inference
NTHU [34]	2020	RRAM	$22nm$	$2Mb$	1-bit	45.5	CIFAR-10	90.18%	Inference

flipping one pixel of the original image. The input signal was represented by the input voltage which was either $+0.1$ V or -0.1 V, corresponding to the black pixel or white pixel respectively. Each weight was implemented by a pair of cells so that the negative weights could be represented. During the in-situ training, the weights to be updated with the same direction, i.e., weight updates (ΔW) that have the same sign, are programmed in parallel, meaning that a set pulse and a reset pulse are applied sequentially to set/reset the corresponding cells. With a proper initialization of the weights, a perfect classification result was achieved after 23 training epochs. Even though the array scale is relatively small and the dataset is quite simple, this pioneering work experimentally demonstrated the feasibility of RRAM crossbar array based neural network training for the first time.

In the same year, Burr et al. from IBM [76] demonstrated a MLP consisting of two hidden layers (with 250 and 125 hidden neurons respectively) and one output layer (10 output neurons) for MNIST handwritten digit classification. The nonlinear activation function was implemented by software while the weighted sum and weight update were measured and implemented with the 500×661 one-transistor-one-resistor (1T1R) PCM array. The weight update during backpropagation was performed using a customized delta rule that sends the stochastic pulses from rows and columns, and the overlap of the two pulses becomes the effective programming window. However, due to the nonlinearity and asymmetry of PCM's conductance tuning characteristics, the training accuracy in this hybrid hardware–software experiments was degraded to $\sim$83%, though the training accuracy could achieve 97% in

the software baseline. To address the issues, an occasional RESET strategy was proposed to mitigate the asymmetry between the gradual conductance increases of PCM partial-SET and the abrupt conductance decrease of a PCM RESET operation, which could be both infrequent and inaccurate. The sensitivity analysis showed that such eNVM-based implementation could be highly resilient to non-ideal random effects (e.g., variability, yield, and stochasticity), but highly sensitive to gradient effects that act to steer all synaptic weights. The simulation results showed that an ideal bidirectional eNVM with a symmetric, linear conductance tuning of high dynamic range would be able to deliver the software-equivalent classification accuracy on the MNIST dataset.

With RRAM, Yao et al. [80] demonstrated a one-layer perceptron for face recognition with a 128×8 analog 1T1R RRAM array. Rather than the unidirectional conductance tuning in PCM, bidirectional analog conductance tuning was achieved in $TaO_x/HfAl_yO_x$ RRAM stack. The one-layer network was trained to recognize and classify grayscale face images from the Yale Face database [85] and tested with the extra unseen faces as well as manipulated images with noisy pixels. For the inference, nine patterns (corresponding to three people) were chosen form the database and cropped and downsampled to 20×16 pixels. Then the inputs were fed to the bitline (BL) as read voltages. The total currents measured on the source line (SL) were measured by the equipment and sent to software-based neuron modules for further processing to generate the final prediction among three classes of faces. For the weight update, two update protocols were proposed: 1) without write–verify which only points out the switching direction depending on the errors sign, following the Manhattan rule; 2) with write–verify which implements both direction and amplitude based on the errors sign and value, following the delta rule. The control circuitry can be much simplified without write-verify but it may slow down the converging speed due to the programming stochasticity. The experimental results showed 91.7% and 87.5% accuracy could be achieved for the scheme with and without write-verify, respectively. The scheme with write-verify shows 4.61X faster converging speed, 1.05X higher classification

accuracy, and 4.41X lower total energy consumption. The network was also tested with random noises in the image pixels and the classification accuracy can be maintained with up to $\sim$31% noisy pixels.

Capitalizing on the progress of low-precision training/inference in the algorithm community, Yu et al. [81] demonstrated a binary neural network (BNN) with a 512×1024 1T1R array. A two-layer MLP was implemented and tested on cropped black-and-white MNIST dataset. For inference, the offline training was performed in software with high precision, then the weights were quantized to 1-bit and programmed to the RRAM array. 96.5% inference accuracy was achieved with binary weights and neurons under non-ideal yield and endurance. This work also explored the case of online training where the weights are updated during run-time. It was shown that at least 6-bit is needed for training to achieve software-equivalent accuracy on MNIST dataset and the endurance became a critical bottleneck that prevented the model from achieving the optimal training accuracy. Considering the immatureness of eNVMs' multi-level programmability, using binary memory cells is more reliable in practice at the expense of larger array size. It should be noted that the proposed methodology was also applicable to other binary memories such as SRAM, PCM, and STT-MRAM.

Though the demonstrations using RRAM array or PCM array as synapses in aforementioned works, the critical neuron circuits, i.e, the discussions on sense amplifiers or ADCs were still missing, meaning that the analog voltages/currents were readout through equipment and post-processed in software. Integrating neuron circuits and possibly more related functionalities (e.g., accumulation, pooling, activation etc.) to form a complete CIM macro naturally becomes the next step. In 2018, Mochida et al. [83] presented a RRAM based design called Resistive Analog Neuro Device (RAND) chip which integrates 4 Mb RRAM synapses and CMOS peripheral circuits including controllers, multiplexers, and sense amplifiers (SAs). Two testchips were fabricated using 180 nm and 40 nm process respectively, the 180nm one achieved 20.7 tera-operations-per-second-per-watt (TOPS/W)

energy-efficiency while the 40 nm one achieved 66.5 TOPS/W. The input data was in the form of 14×14 pixel images by compressing the MNIST dataset. A circuit for searching the max value of MAC operation was designed to deal with the sensing offsets and current variations, which improved the inference accuracy from 87.3% to 90.8%. However, since only one SA was utilized to generate 1-bit neuron output, there was still a notable gap compared to the software baseline. To this end, Chen et al. [32] presented a 65 nm RRAM based binary-input ternary-weight MLP with optimized 3-bit distance-racing current-mode sense amplifiers (CSAs). An input-aware dynamic reference generation scheme was proposed that could dynamically generate the reference currents through a reference array according to the number of "1"s in the input vector. With the 3-bit neuron output and suppressed sensing offsets, a two-layer MLP was demonstrated, achieving 96.2% on MNIST dataset, and it was claimed that the accuracy could achieve $>$98% if using a five-layer MLP.

Nevertheless, MNIST dataset is still a toy task compared to the real-world AI applications, since then, the demonstration with more complex datasets is of more interests. In 2019, Xue et al. [33] improved their previous design [32] to support up to 2-bit input, 3-bit weight, and 4-bit neuron output. A RRAM testchip was fabricated using 55 nm process that implemented a CNN targeting CIFAR-10 dataset. This work proposed a serial-input non-weighted product scheme, where the non-weighted current was accumulated within the array and designated peripheral circuits were used to perform the summation with different significances. A triple-margin CSA that could achieve up to 6X offset suppression compared to traditional current-latch CSA was deployed to improve the read yield. The measurement results showed 81.8% accuracy on CIFAR-10 and 53.2 TOPS/W energy-efficiency with the configuration of 1-bit input, 3-bit weight, and 4-bit neuron output. One limitation to note here is that only 9 rows were activated simultaneously during MAC operation which undermined the throughput and energy-efficiency of the system. To fully leverage the potential of the parallelism of CIM architecture, it's preferred to assert all the rows at one time. Yin et al. [36] designed a 90 nm RRAM testchip that integrated a 128×64

array and a group of 3-bit Flash ADCs for implementing binary CNN operations, where all the rows were asserted simultaneously to maximize the parallelism. The resistance distribution of low resistance states (LRS) was tightened by write-verify technique, and the ADC reference voltages were calibrated to mitigate the impact of offset. The measurements showed 98.5% for MNIST dataset and 83.5% for CIFAR-10 dataset with energy efficiency of 24 TOPS/W and 158 GOPS throughput, which means 5.6X and 3.2X improvements in throughput and energy-delay product (EDP) respectively compared to [33].

In 2020, Xue et al. [34] further extended [33] to support up to 4-bit input, 4-bit weight, and 11-bit output, while achieving 28.9 TOPS/W. It only showed 0.93% accuracy degradation when evaluated on CIFAR-100 dataset [51]. It should also be noted that when configured to 1-bit input, 2-bit weight, and 6-bit output, this design could achieve 121.3 TOPS/W as the best case.

CHAPTER 3

RRAM-BASED INFERENCE CHIP DESIGN

3.1 Introduction

As we have pointed out in Chapter 1, SRAM is commonly utilized to store the weights of DNNs in CMOS ASIC accelerators. However, a SRAM cell consumes $> 150F^2$ (F = technology feature size) in terms of area, which limits the storage capacity of on-chip weights. Considering the substantial amount of weights in state-of-the-art DNNs (e.g., ResNet-152 [3] contains $\sim$60M parameters), it is prohibitive to directly implement the DNN model on-chip with SRAM. Therefore, eNVMs draw lots of interest in the recent years as a promising candidate for on-chip weight storage due to the much higher cell density. Meanwhile, with the highly parallel analog computing paradigm of CIM, the intensive data movements between the processor and memory can be significantly reduced, leading to the improvements on throughput and energy-efficiency. As discussed in Chapter 2, RRAM, PCM, and FeFET are the most representative eNVM technologies that have shown great promises for both conventional memory application and CIM application. Even though there are literatures showing potential multi-bit per cell programmability, the non-ideal analog cell characteristics (e.g., non-linear conductance tuning, limited dynamic range, conductance variation etc.) could introduce significant accuracy degradation. Hence, it is more practical to leverage the technologically more mature binary eNVMs that have shown industry-mature Gb-level demonstration as the near-term solution.

In this chapter, we present two generations of RRAM-based testchips taped-out using commercial RRAM process, namely XNOR-RRAM (Winbond 90 nm process) and Flex-RRAM (TSMC 40nm process) that demonstrate the feasibility of CIM operation for accelerating the inference of DNNs. XNOR-RRAM chip is dedicated for binary neural net-

works where weights and activations are quantized to binary state, and Flex-RRAM chip can support 1bit-to-8bit flexible precision configurability at run-time. Flex-RRAM also features adaptive input sparsity control which takes advantage of the high sparsity nature of typical DNN models to improve the throughput and energy-efficiency. In addition, Flex-RRAM also supports on-chip input-aware reference generation that provides fine-grained tunability of ADC reference voltages, and on-chip write-verify control that can speed up the weight programming phase drastically. The measurement result of XNOR-RRAM chip shows that it can achieve energy-efficiency of 24.1 TOPS/W at 1.2V while maintaining a acceptable classification accuracy on CIFAR-10 dataset. As the Flex-RRAM chip is still in fabrication at foundry, the post-layout simulation result shows that it could achieve 38.6 TOPS/W (when configured for BNN) at 0.9V assuming the sparsity of input activations is 50%.

The rest of this chapter is organized as follows: Section 2 briefly introduces the basic information of RRAM technology. Section 3 and Section 4 present the design of XNOR-RRAM chip and Flex-RRAM chip respectively. Section 5 summarizes the chapter.

3.2 RRAM Basics

RRAM is a non-volatile two-terminal device technology where the cell resistance can transition between the high resistance state (HRS) and the low resistance state (LRS). Generally, the resistance is increased and decreased by applying voltage pulses with different polarities, which is referred to as RESET and SET process respectively. There are typically two main categories of switching mechanisms of RRAM devices: one is the so-called filamentary mechanism which relies on the formation and rupture of the conductive filament that consists of metal ions or oxygen vacancies [64, 65, 66, 67, 68]; the other one is based on the interfacial mechanism where the distribution of oxygen vacancies at the interface is modulated by the electric field [69, 70, 71, 72]. Filamentary RRAM is widely adopted for digital memory application for its fast and low-power switching process. However,

the formation of the filament, i.e., the SET process, is typically abrupt due to the positive feedback between the growth of the filament and the electric field. Thus, a unidirectional RESET-only updating scheme was adopted in the HfO_x-based synaptic device [73]. In order to make the SET process gradual, bilayer-oxide structures (e.g., TaO_x/HfO_2 [74] and AlO_x/HfO_2 [75]) have been proposed to form multiple weak filaments instead of one single strong filament. On the other hand, the interficial RRAM shows gradual SET and RESET process. However, the resistance tuning curve typically suffers from nonlinearity and asymmetry, thus requiring more complex programming schemes with non-identical pulses for mitigation.

The requirements on device characteristics are different depending on the application scenarios, i.e., for in-situ training or for inference. For inference, the write-verify technique can be utilized to iteratively program the RRAM cell to the desired resistance level. Since this programming process is one-time only before the deployment of the chip, the complexity, speed, and energy consumption of the programming process are not top concerns as long as the resistance level can be accurately programmed. On the other hand, for in-situ training, the requirements on programming are very critical because the training of DNNs typically involves a huge iterations of weight updates in both directions, which means the write-verify may not be feasI_{BL}e in this case due to the substantial latency and energy overhead. Therefore, a linear, symmetric, and accurate resistance tuning without write-verify is preferred.

In this chapter, we design two generations of RRAM-based inference testchips with commercial RRAM processes that are originally developed for embedded memory application. Thus, we only use the binary RRAM cell to implement weights. For the first generation XNOR-RRAM chip, which is dedicated for BNNs, the binary cell is naturally a perfect fit for implementing binary weights. For the second generation Flex-RRAM chip, which supports up to 8-bit weight precision, we use multiple cells in the same row to represent one weight.

3.3 XNOR-RRAM Prototype Chip: RRAM-based CIM for Binary Neural Networks

3.3.1 Binary Neural Networks

Table 3.1: Classification Accuracy in Different Cases

Network	Dataset	FL Precision	Binary Precision
MLP	MNIST	99.00%	98.77%
CNN	CIFAR-10	89.98%	88.47%

In binary neural networks, both the weights and neuron activations are binarized to -1 or +1. Therefore, multiplications between activations and weights can be simplified as XNOR operations and accumulation of the products is equivalent to bit-counting operation. In this paper, we trained BNNs using the algorithm proposed in [86] on the Theano platform. A multilayer perceptron (MLP) with a structure of 784-512-512-512-10 and a convolutional neural network (CNN) with 6 convolution layers and 3 fully-connected layers are trained for evaluations on MNIST and CIFAR-10 datasets, respectively. Table 3.1 presents the corresponding classification accuracy with floating point (FL) precision and binary precision for these two networks. For MLP on MNIST, the accuracy slightly drops from 99.00% to 98.77%; for CNN on CIFAR-10, the accuracy slightly decreases from 89.98% to 88.47%. Such minor degradations have also been observed in state-of-the-art BNN algorithms [86, 87].

3.3.2 XNOR-RRAM Architecture with Customized Weight Cell

Figure 3.1(a) presents the principle of the proposed bit-cell design for XNOR-RRAM implementation. For each synaptic weight, "-1" is represented by two cells where the top one is in HRS and the bottom one is in LRS. The reversed pattern is used for "+1". For the wordline (WL) input pattern, two adjacent WLs for each weight-cell are in complimentary state where (0, 1) represents "-1" and (1, 0) represents "+1". In this way, the amount of the current that flows through each weight-cell during read-out is dependent on the com-

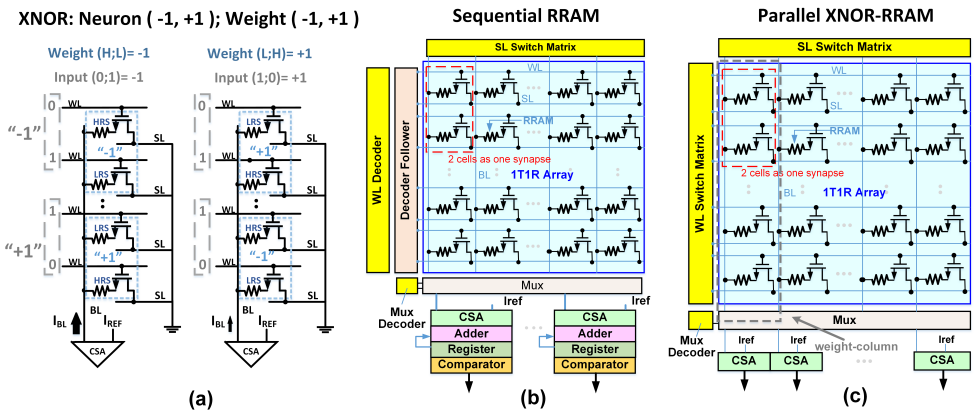


Figure 3.1: (a) The customized bit-cell design for XNOR implementation. (b) The diagram of conventional sequential RRAM synaptic architecture. (c) The diagram of proposed parallel XNOR-RRAM architecture.

bination of WL input pattern and bit-cell pattern. For example, when input vector is "-1", for the cell of weight "-1", the cell in the activated row is in LRS, resulting in a large cell current, which can be regarded as a bit-wise XNOR output of "+1". For the cell of weight "+1", the cell in the activated row is in HRS, leading to a small cell current, which can be regarded as a bit-wise XNOR output of "-1". When multiple WLs are activated in parallel, the LRS-cells will dominate the total bit line current (I_{BL}) if the on/off ratio of RRAM is sufficiently large. Consequently, I_{BL} will be proportional to the bit-counting results equivalent to the number of LRS-cells along the column. For example, 50% "+1" and 50% "-1" will lead to a final weighted sum of 0. Assuming the column length of the sub-array is 64, the sum of 0 can be mapped to the I_{BL} = 32 activated LRS-cells. Therefore, the reference current (I_{REF}) for the current sense amplifier (CSA) could be set to 32 LRS-cells' current for the binary neuron activation. If I_{BL} is smaller than I_{REF} that generates a CSA output "-1", it represents that there are more "-1" than "+1" along the column, and vice versa.

For the sequential RRAM design in Figure 3.1(b), the encoded input neuron vector is fed into WL decoder, and only one WL is activated in each read-out operation. During the read-out operation, V_{BL} is biased to be ground, CSA imposes current on the selected BL and compares I_{BL} with the fixed I_{REF} to determine the output. For each weight column,

there are MAC units such as adder and register pair at the end of the column for row-by-row summation and partial weighted sum storage. In the end, the final weighted sum goes through a digital comparator to generate 1-bit neuron output (+1 or -1). For the parallel XNOR-RRAM design in Figure 3.1(c), instead of a normal decoder, a WL switch matrix is designed to activate multiple WLs simultaneously depending on the input vectors to enable the parallel read-out operation. The parallel XNOR-RRAM architecture leverages the analog current summation to effectively realize the MAC operation, thus the adder/register periphery of the sequential row-by-row scheme is eliminated.

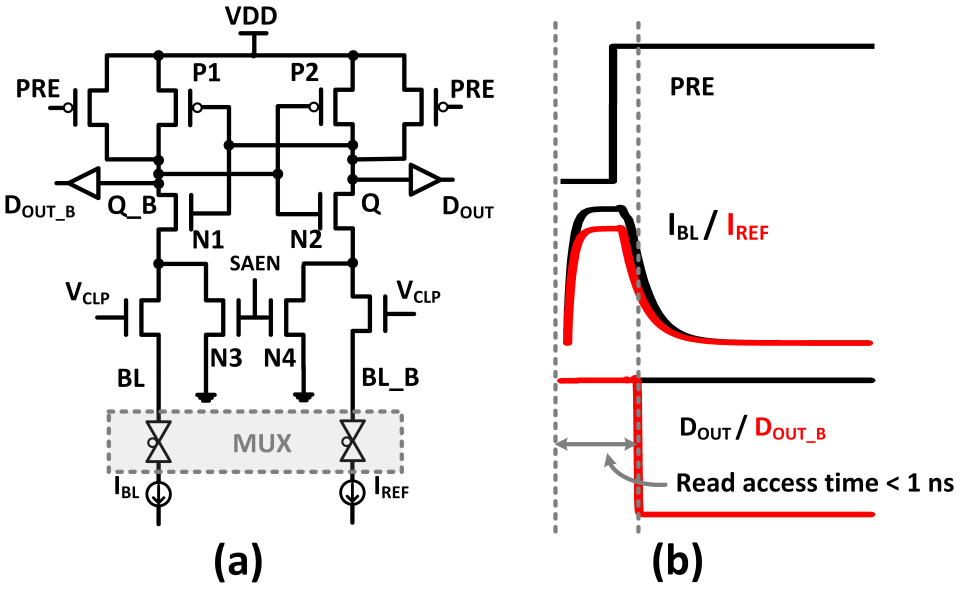


Figure 3.2: (a) Schematic of CL-CSA. (b) Simulation waveforms of sensing 40 LRS-cells (BL) against 32 LRS-cells (BL_B). As I_{BL} is larger than I_{REF}, D_{OUT} remains as "1" while D_{OUT_B} drops to "0". Read access time is less than 1 ns.

Theoretically, a 1-bit CSA can serve as the binary neuron for each column in parallel XNOR-RRAM to generate the binary neuron output. However, due to the intrinsic offset of CSA, the sensing pass rate (percentage of accurate sensing results) becomes worse when I_{BL} increases (as cell currents are summed up for a large array), which may bring significant accuracy degradation as the threshold of the neuron may differ from the ideal value in algorithms. In this section, we perform Monte Carlo (MC) simulations to investigate the

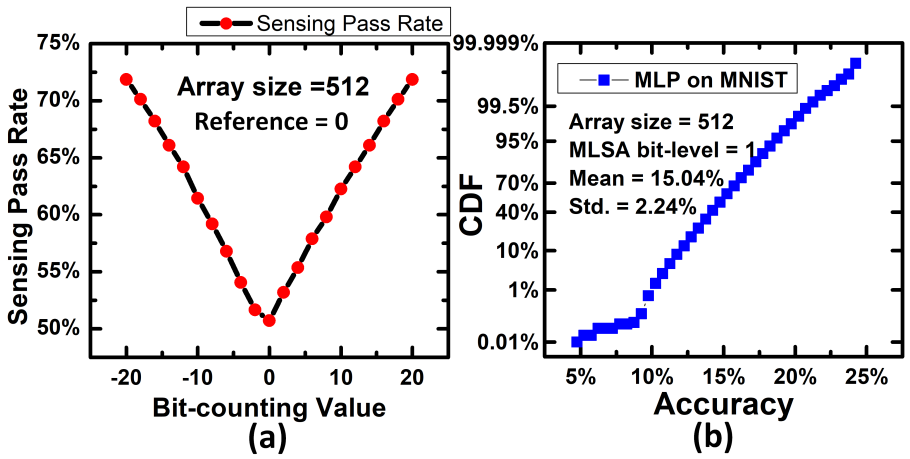


Figure 3.3: (a) The sensing pass rate of different bit-counting values when the array size is 512 $\times$ 512. The reference is set as the current for the bit-counting value of 0. (b) The accuracy distribution of MLP on MNIST from 10,000 runs where one single CSA is used as the binary neuron. The average accuracy is only 15.04%.

impact of 1-bit CSA offset on classification accuracy of the MLP, using a 512 $\times$ 512 RRAM array. A current-latch based CSA [88] is employed, comprising precharge PMOS, cross-coupled pair, and pull-down NMOS as shown in Figure 3.2(a). During the precharge phase, CSA imposes large precharge current to raise the voltage on BL/BL_B to drive I_{BL} and I_{REF}. When the difference between I_{BL} and I_{REF} reaches its maximum, the precharge transistors turn off and the cross-coupled pair compares the current difference to determine the output value. The offset of CSA is mainly due to the trip-point voltage mismatch between P1-N1 and P2-N2 that is caused by process variation. In the simulation setup, LRS/HRS resistance is assumed to be 200KΩ / 200MΩ. The waveform in Figure 3.2(b) shows the case of sensing 40 LRS-cells (BL) against 32 LRS-cells (BL_B) as an example. As I_{BL} is larger than I_{REF}, the voltage at node Q_B drops to the trip-point voltage earlier than node Q, raising node Q toward VDD. As a result, D_{OUT} remains at VDD while D_{OUT_B} drops to "0". Since I_{BL}/I_{REF} become much larger due to parallel read-out, the read access time is observed to be less than 1 ns. As aforementioned, the bit-counting results can be mapped to different number of activated LRS-cells in the corresponding column, hence the I_{BL} with 256 activated LSR-cells represents a sum of 0 in this case (for a 512 $\times$ 512 array

size). For the illustration purpose, here we only perform 21 sets of simulation covering bit-counting results from "-20" to "+20". For each set, 10,000 MC points are simulated by Cadence Spectre using TSMC 65nm PDK. Figure 3.3(a) shows the sensing pass rate of different bit-counting values, where sensing failures may occur due to the offset. Even with "-20" or "+20" difference in the bit-counting value, the sensing pass rate is less than 80%. As there are $512 + 512 + 10 = 1,034$ binary columns in total for the MLP, every 1,034 MC points are randomly selected as one group each time to generate 10,000 groups of offset patterns. Then we perform the inference on MNIST dataset with the generated offset patterns. Figure 3.3(b) shows the distribution of the classification accuracy from 10,000 MC runs. The average accuracy is only 15.04%, which is definitely insufficient to achieve a good accuracy. Therefore, we propose to split a large weight matrix into small ones to maintain a good sensing pass rate of CSA.

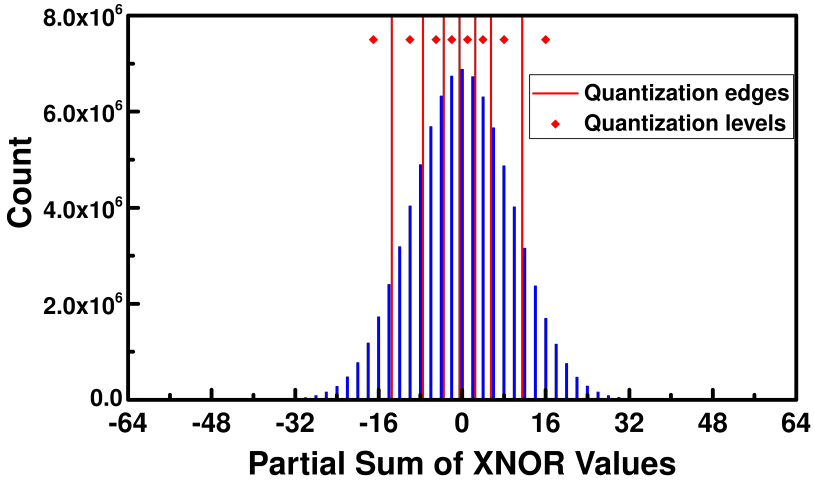


Figure 3.4: Distribution of partial sums of XNOR values of the MLP. Sub-arrays are assumed to be 64×64. Red lines are 7 nonlinear quantization edges (or references) and red diamonds indicate 8 quantization levels.

The size of the sub-array is a key design parameter that affects the classification accuracy and hardware overhead of system. After the matrix splitting, each small matrix needs to generate a partial sum, which will be added up to obtain the final sum for binary activation. Thus, the precision of the partial sum will affect the value of the final sum thus

determining the classification accuracy. As a result, ADCs are employed to generate partial sums with fixed-point precision (larger than 1-bit). To minimize the quantization error of the partial sums, we propose to perform nonlinear quantization where quantization edges (or references) are determined via Lloyd-Max algorithm [89] according to the distribution of the partial sums. For instance, the distribution of the partial sums in the evaluated MLP is shown in Figure 3.4. 7 quantization edges (or references), and 8 quantization levels acquired from Lloyd-Max algorithm are also annotated. Due to reduced quantization error, nonlinear quantization achieves better accuracy than linear quantization, given the same number of quantization levels. For example, the CNN for CIFAR-10 achieves test accuracy of 86.68% with nonlinear quantization and only 13.90% with linear quantization for 8 quantization levels (or 3-bit ADCs).

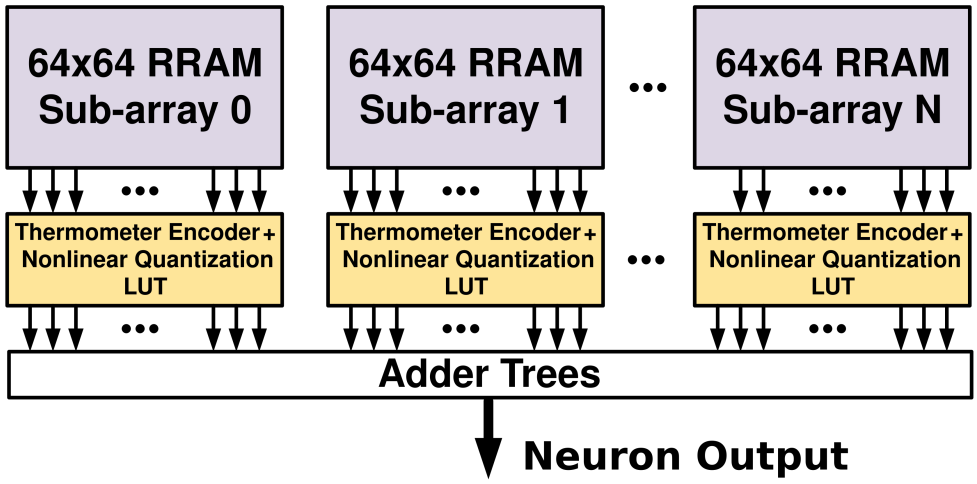


Figure 3.5: Generic system diagram for implementing one layer with arbitrary size in a network. The size of sub-array is assumed to be 64×64 as an example.

A generic system diagram that implements one BNN layer of arbitrary size is shown in Figure 3.5 (sub-arrays are assumed to be 64×64).ADCs in sub-arrays generate digital outputs with fixed-point precision, which then go through a thermometer encoder and look-up table (LUT) to be converted to the corresponding quantization values as partial sums. Adder trees sum up the partial sums to be the final weighted sum, which then goes through the binary activation to generate the neuron output. Here we investigate the cases for the

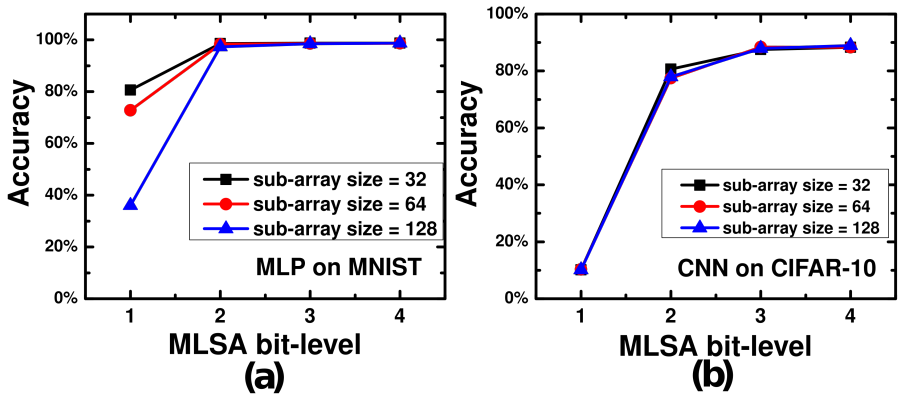


Figure 3.6: The classification accuracy of different sub-array sizes and MLSA bit-levels for (a) MLP on MNIST and (b) CNN on CIFAR-10. A 3-bit MLSA is sufficient to provide satisfying accuracy for both evaluations.

array size of 32 $\times$ 32, 64 $\times$ 64, and 128 $\times$ 128 with ADC precision ranging from 1-bit to 4-bit (i.e., 2, 4, 8, 16 quantization levels). The software simulation results for each case are shown in Figure 3.6. It can be observed that for both MLP and CNN with 3 different sub-array sizes, the classification accuracy saturates when ADC precision reaches 3-bit. In the meantime, 2-bit ADCs can also provide $> 98\%$ accuracy (degradation of $< 1\%$) for MLP on MNIST when sub-array size is 32 $\times$ 32 or 64 $\times$ 64.

3.3.3 XNOR-RRAM Prototype Chip Design

Based on the observation in the previous section, we finalize the sub-array size to be 128×128 and the ADC precision to be 3-bit. A prototype chip is fabricated with Winbond's embedded RRAM technology [37], which monolithically integrates 90nm CMOS and RRAM between M1 and M2. The micrograph of the chip is shown in Figure 3.7(a), the chip size is $5mm \times 5mm$, and Figure 3.7(b) shows the core area of the XNOR-RRAM chip. As shown in the top-level block diagram in Figure 3.7(c), the chip consists of a 128×64 1T1R array that effectively represents a 64×64 weight matrix, 8-to-1 column multiplexers, 3-bit flash ADCs, row decoder, level shifters, and column decoders. Due to the area mismatch between the RRAM array and flash ADCs, every 8 RRAM columns

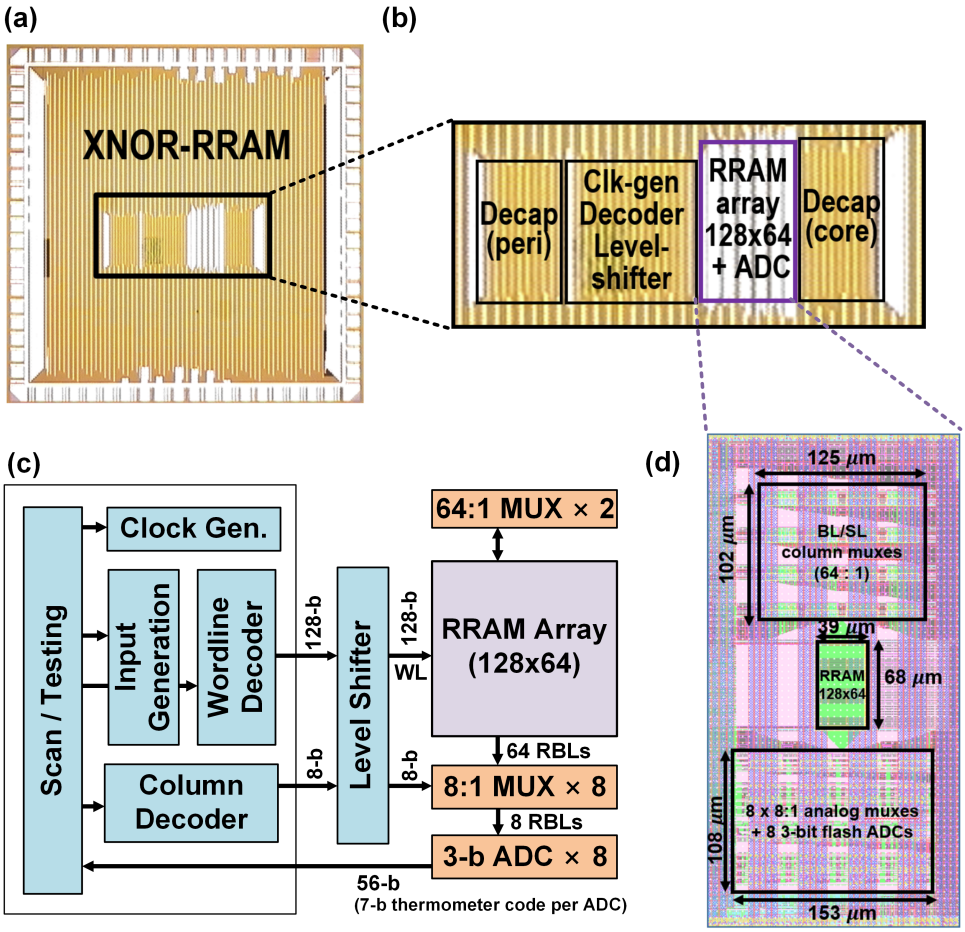


Figure 3.7: Overview of the XNOR-RRAM prototype chip fabricated with Winbond 90nm RRAM technology. (a) The micrograph of XNOR-RRAM chip. Die size is $5mm \times 5mm$. (b) The core of the XNOR-RRAM chip, consisting of RRAM array, MUXs, ADCs, decoders, and level-shifters. (c) Top-level diagram of XNOR-RRAM chip design. (d) Dimensions of the RRAM array, column multiplexers and flash ADCs. Note that this is a collaborated project with ASU and the author was responsible for all the custom-designed blocks including RRAM array and analog peripheral circuits in this tape-out. ASU collaborators were responsible for digital synthesized modules and top-level integration. Adapted from [90].

share one 3-bit flash ADC. As a prototype chip, we focus on the demonstration of the essential CIM operation other than the ADC design. Therefore, we use common VSAs in the chip design instead of CSAs adopted in the previous simulation work [91] as providing voltage references externally is much more straightforward than providing current references. Moreover, current mirrors are needed to replicate the current references to 8 ADCs, which are suffering from mismatches caused by process variations, thus leading

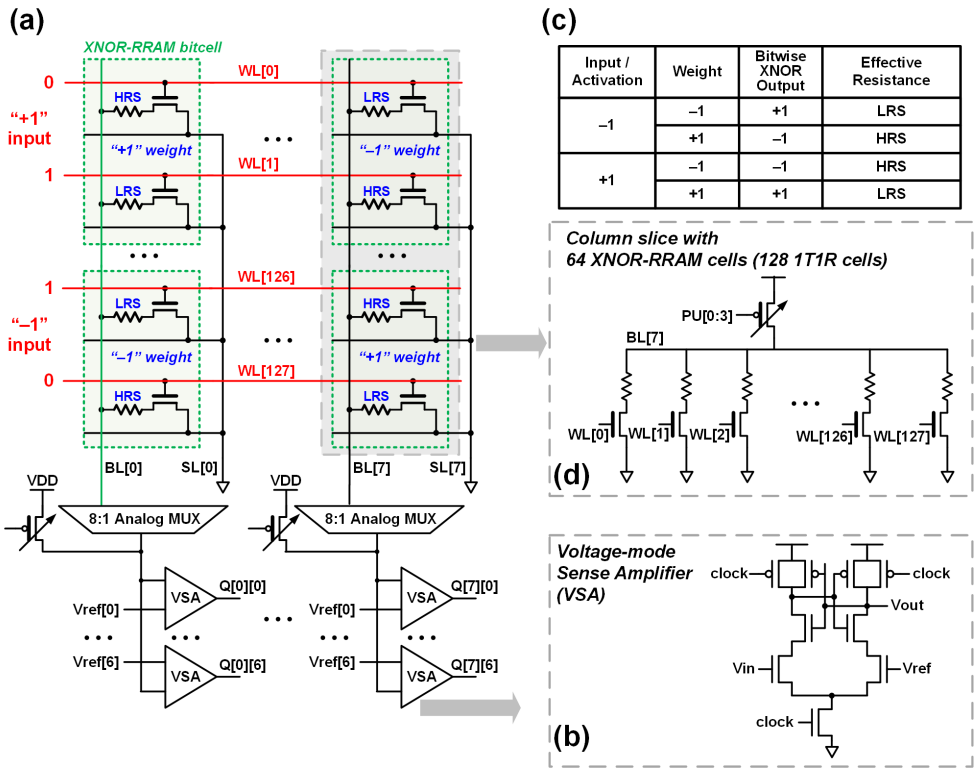


Figure 3.8: The working principle of XNOR-RRAM chip. (a) XNOR-RRAM macro architecture. (b) The schematic of the VSA, which compares V_{IN} and V_{REF} at the rising edge of the clock. (c) Truth-table of the equivalent XNOR operation. (d) The voltage-divider structure formed between the static pull-up PMOS and the pull-down network with RRAM cells along the column. Adapted from [90].

to the errors in references. The row decoder has two operating modes for read-out and programming respectively. During read-out operation, i.e., to perform weighted-sum CIM operation for inference, all differential WL signals are activated simultaneously. During weight programming, it generates one-hot WL signals for cell-by-cell programming as a conventional address decoder. The area of the 1T1R bitcell is around $0.5\,\mu\text{m} \times 0.5\,\mu\text{m}$ ($31F^2$, where F is technology node). Flash ADCs and column multiplexers consume 20% and 12% area of the core area, respectively (Figure 3.7(d)).Please note this is a collaborated work with Arizona State University (ASU), and the author was responsible for all the custom-designed blocks including RRAM array and analog peripheral circuits in this tape-out. ASU collaborators were responsible for digital synthesized modules and top-level integration. The measurement was conducted at ASU by our collaborators.

Figure 3.8 shows the principle of the CIM operation of XNOR-RRAM. As shown in Figure 3.8(a), the weights are represented by deferentially programmed RRAM cells and the input activations are encoded by differential wordline signals. Each 3-bit flash ADC consists of 7 voltage-mode sense amplifiers (Figure 3.8(b)). Figure 3.8(c) presents the truth-table of the embedded XNOR function where the output is represented by the corresponding effective cell resistance. Figure 3.8(d) shows the voltage-divider structure exploited to generate the eventual bitline voltage to be sent to ADCs for conversion. A static PMOS header, the strength of which is configurable, pulls up the RBL voltage. The RRAM cells along the same column pull down the RBL voltage in parallel. Depending on the activated cell pattern in the column, a static resistive divider is formed between the pull-up PMOS and the pull-down RRAM cells. For example, if more LRS RRAM cells are activated (higher bitcount value from the algorithm), RBL voltage will be lower. Thus, different bitcount values are mapped to different RBL voltage levels.

3.3.4 XNOR-RRAM Chip Measurement Results

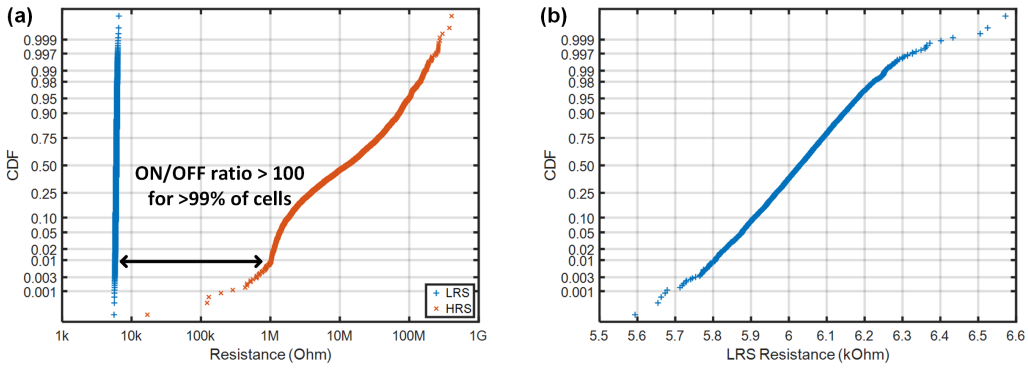


Figure 3.9: The programmed resistance distribution of RRAM devices. (a) Resistance distribution of 4,096 LRS cells and 4,096 HRS cells. (b) The tightened LRS resistance distribution near 6 kΩ through write-verify. Note that the measurement of XNOR-RRAM chip was conducted at ASU by our collaborators. Adapted from [90].

The measurement of XNOR-RRAM chip was conducted at ASU by our collaborators. Figure 3.9 shows the testchip measurement results of LRS and HRS distribution for the 128×64 array. The LRS distribution is much tightened near the suggested LRS level 6 kΩ

while the HRS distribution is relatively more relaxed. > 100 ON/OFF ratio is achieved for $> 99\%$ of RRAM cells. Tightening LRS distribution is more important for CIM operation as the column current will be dominated by the current through LRS cells. To achieve the shown distribution, we apply an amplitude-modulated write-verify scheme. The target resistance range is set as 5.9 kΩ to 6.1 kΩ. The initial gate voltage is set to 2.3 V and a 100 ns SET pulse is applied with amplitude of 2.1 V. If the resistance after SET pulse is lower than the lower bound, a 200 ns RESET pulse with amplitude of 3.8 V and gate voltage of 4 V is applied to the RRAM cell to fully reset the cell to HRS, then followed by a SET pulse with a 0.05 V lower gate voltage. If the resistance after SET is higher than the upper bound, a RESET pulse is applied to the cell followed by a SET pulse with a 0.05 V higher gate voltage. The iteration limit is set to 10, meaning that we will move to the next cell if the resistance of the current cell is not pushed to the desired range in 10 iterations. As for the programming of HRS, the target HRS resistance value is set to be above 1 MΩ. We keep applying a 200 ns RESET pulse with amplitude of 3.8 V and gate voltage of 4 V to the cell for up to 10 iterations. The resistance values are read at 0.2 V by a source measurement unit (SMU).

Figure 3.10 shows the measurement results of ADC. We perform the measurement with two different configurations: (1) only one common set of reference voltages is used for all the 8 ADCs and (2) Each set of reference voltages is calibrated for each ADC, meaning that the sensing offset is eliminated. The RRAM array is firstly programmed based on the trained binary neural network for MNIST using the aforementioned write-verify scheme, then 2,000 64-bit input vectors are fed to the chip and the corresponding ADC outputs are collected. In total, 128,000 pairs of measured ADC outputs and the ideal partial sum values are used to analyze the effectiveness of the CIM operation. As shown in Figure 3.10 (a) and (c), the bitcount value and the ADC output show an expected linear relationship while the case with 8 calibrated sets of reference voltages presents much tighter distribution, which means less error in partial sum results. This can be further observed in (b) and (d) where

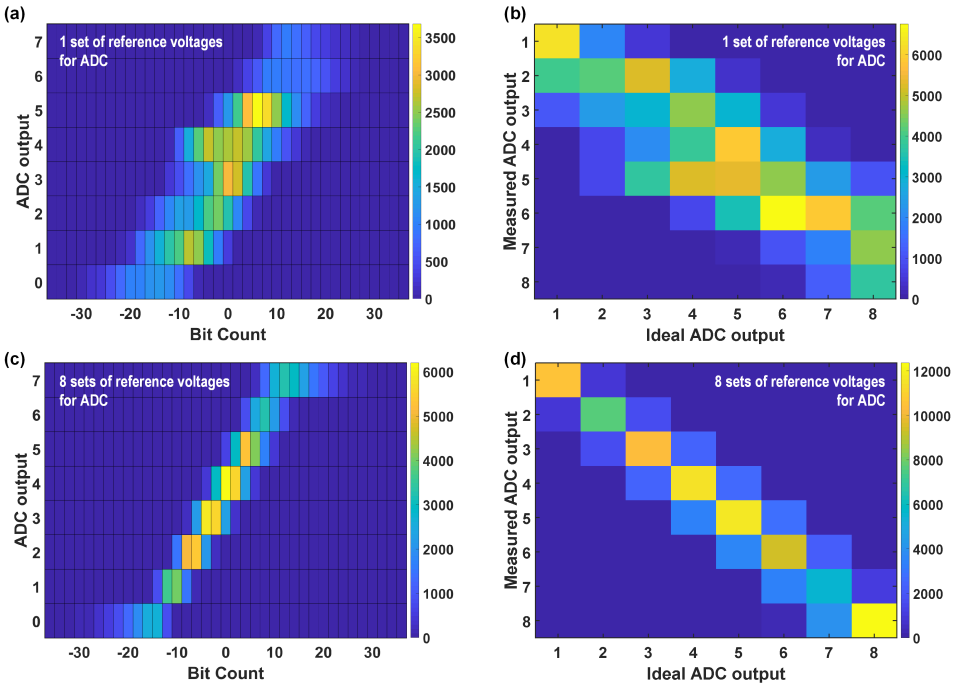


Figure 3.10: ADC measurement results. (a) and (b) shows the case where 1 common set of reference voltages is used for all ADCs. (c) and (d) show the case where the reference voltages are calibrated for each ADC, meaning that the sensing offset is eliminated. Adapted from [90].

we compare the measured ADC output with the ideal ADC output based on the partial sum quantization in algorithm and all the points should lie on the diagonal of the plot in the ideal case. We can see that (d) bears much less deviations compared to (b). If the VSA/ADC is enhanced with offset cancellation circuits [92], then all ADCs can share the same set of reference voltages, making it more viable in practice.

3.3.5 DNN Accuracy Evaluation

We evaluate the accuracy of XNOR-RRAM chip for implementing realistic deep neural networks for MNIST and CIFAR-10 datasets through a software-measurement combined methodology. As shown in (Figure 3.11, for MNIST, we use a MLP with 3 hidden layers, each hidden layer consists of 512 neurons. For CIFAR-10, we use a variant network based on VGG-Net, which consists of 6 convolution layers and 3 fully-connected layers [93]. As our XNOR-chip only implements weighted-sum operation, the batch-normalization layers

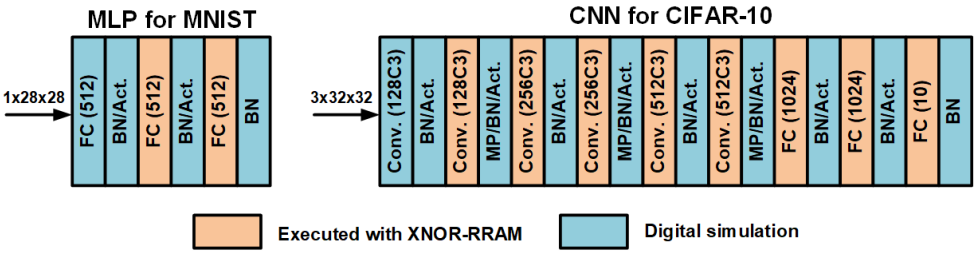


Figure 3.11: Evaluation of DNNs for MNIST and CIFAR-10 datasets using XNOR-RRAM chip. A MLP with a structure of 784-512-512-512-10 is used for MNIST dataset. A variant of VGG-Net, namely, VGG-9 with 6 convolution layers and 3 fully-connected layer is used for CIFAR-10 dataset. Adapted from [90].

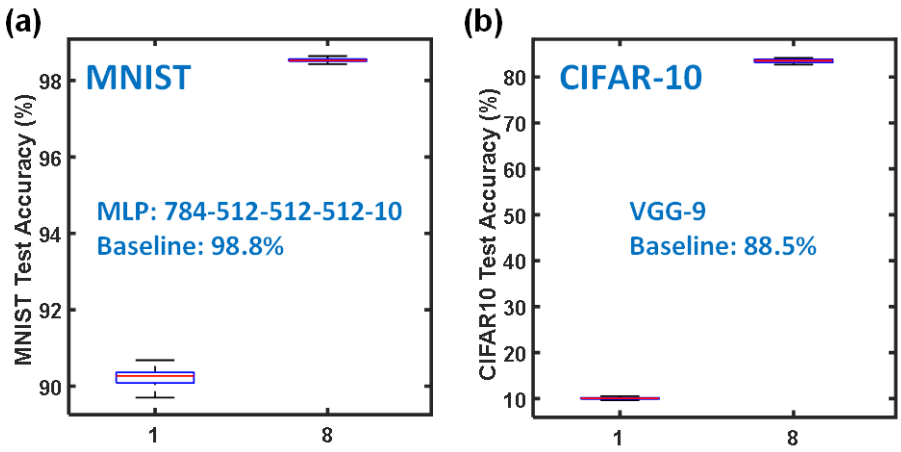


Figure 3.12: Accuracy results of (a) MLP on MNIST and (b) CNN on CIFAR-10 from the software-measurement combined evaluation. The significant accuracy improvement from using 1 set to 8 sets of reference voltages shows that offset cancellation is crucial. Adapted from [90].

and activation functions are performed in software. Figure 3.12 shows the accuracy results with the aforementioned two different reference voltage schemes respectively. Accuracy numbers are obtained from 20 runs, where the partial sums in each run are stochastically quantized based on the probability distribution in Figure 3.10. As expected, using 8 independent sets of reference voltages for 8 ADCs shows significant accuracy improvement in both MNIST and CIFAR-10 evaluation compared to the scheme with a single set of reference voltages. Using 8 independent sets of reference voltages, our XNOR-RRAM chip achieves 98.5% classification accuracy for MNIST dataset (software baseline: 98.8%), and 83.5% classification accuracy for CIFAR-10 dataset (software baseline: 88.5%). The ac-

curacy degradations are due to limited ADC precision that fails to correctly distinguish two adjacent reference levels in some cases, which could be mitigated by using an ADC with higher precision [94] (trading off ADC area and power) or activating less number of rows [33] simultaneously to relax the requirement no sensing margin (trading off latency or energy-efficiency).

3.3.6 Power, Energy, and Throughput Results

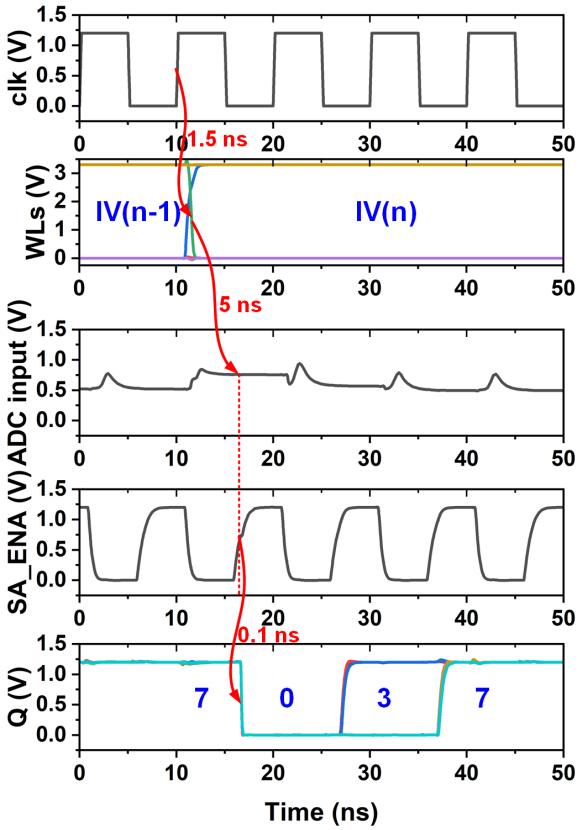


Figure 3.13: Each input vector is presented to XNOR-RRAM array for 8 cycles. Each ADC senses the RBL voltage of one of the 8 columns each cycle. From top to bottom: global clock signal, word lines driving the XNOR-RRAM array, RBL voltage as the ADC input, ADC clock signal (sense amplifier enable signal), flash ADC outputs. The clock to word line delay, RBL voltage settling delay, and flash ADC sensing delay are 1.5 ns, 5 ns and 0.1 ns, respectively. Adapted from [90].

It should be noted that the clock frequency in our measurement is limited by the slow IO pads at $\sim$ 20 MHz. However, to achieve the best energy-efficiency performance, we

should run the chip at the maximum frequency, which is determined by the bitline settling time. Therefore, we report the post-layout simulation results as shown in Figure 3.13, the critical path from clock rising edge to the settling of bitline voltage is 6.5 ns, meaning that the chip could operate at most 154 MHz. With this assumption, the chip could achieve 24.1 TOPS/W at 1.2 V, which could be further improved by scaling down the power supply. However, the ADC sensing margin will be reduced as well if the power supply is scaled, leading to accuracy degradation on classification accuracy. For example, the accuracy drops to 97.28% on MNIST dataset when power supply is scaled to 0.9 V. As for throughput, XNOR-RRAM chip (with one 128×64 array) achieves a high throughput of 157.7 GOPS, which can be attributed to the assertion of 128 rows in parallel.

Table 3.2: Comparison between XNOR-RRAM Chip and the Prior Work

Metric	NTHU [33]	XNOR-RRAM
CMOS Technology	55 nm	90 nm
Sub-array Size	256×512	128×64
Nominal VDD	1 V	1.2 V
Precision (bits)	A:1/W:Ternary/O:4	A:1/W:1/O:3
Energy-Efficiency (TOPS/W)	53.17	24.1
Throughput (GOPS)	7.1	157.6
FoM	375.4	3798.2
CIFAR-10 Accuracy	81.83%	83.50%

Table 3.2 shows the comparison with the prior work implemented in 55nm RRAM process [33]. [33] only turns 9 rows simultaneously during CIM operation while our work turns on all 128 rows of the RRAM array in parallel, leading to 22.3X higher throughput per 128×64 array macro operation. Considering the widely adopted metric energy-delay product (EDP), we use the figure-of-merit (FoM), which is the product of energy-efficiency (TOPS/W) and throughput (GOPS), to effectively represent the inverse of EDP. This work achieves 10.1X better performance compared to the prior work. Meanwhile, we achieve a better accuracy on CIFAR-10 dataset thanks to the confined-range linear quantization and the ADC reference calibration.

3.4 Flex-RRAM Prototype Chip: RRAM-based CIM with Configurable Precision

3.4.1 Flex-RRAM Prototype Chip Overview

Despite the savings on area and energy overhead, BNNs typically fail to achieve the baseline accuracy with floating-point precision, especially when applied to more complex tasks, such as ImageNet classification and speech recognition etc. Thus, it is of great interest to enable the CIM operation with multi-bit precision. In this section, we present the second generation of our RRAM-based inference chip designed with TSMC 40 nm RRAM process, namely Flex-RRAM chip, which supports 1bit-to-8-bit configurable precision at run-time. Flex-RRAM also features adaptive input sparsity control which takes advantage of the high sparsity nature of typical DNN models to improve the throughput and energy-efficiency. In addition, Flex-RRAM supports on-chip input-aware reference generation that provides fine-grained tunability of ADC reference voltages, and on-chip write-verify control that can largely speed up the weight programming phase.

The top-level diagram of Flex-RRAM chip is shown in Figure 3.14. The blocks in grey are custom-designed analog blocks. The structure of the ADC is the same with XNOR-RRAM chip, which is 3-bit flash ADC consisting of 7 voltage-mode sense amplifiers (VSA). Level-shifters are used to bridge the voltage gap between the logic control (0.9V) and the higher voltages required for SET/RESET/Forming (up to 3.2V). To speed up the write-verify process during weight programming, we implement a on-chip write-verify module instead of using testing equipment as what had been done for XNOR-RRAM chip. A dual-SA module is designed with large I/O transistors to suppress the offset caused by process variations. In addition, a RRAM-based input-aware reference generation module is implemented with a portion of the RRAM array, which can be fine-tuned to calibrate the level of references.

The modules in white are synthesized digital modules. A sparsity-aware adaptive row-input control module is designed to dynamically assert a certain number of rows while

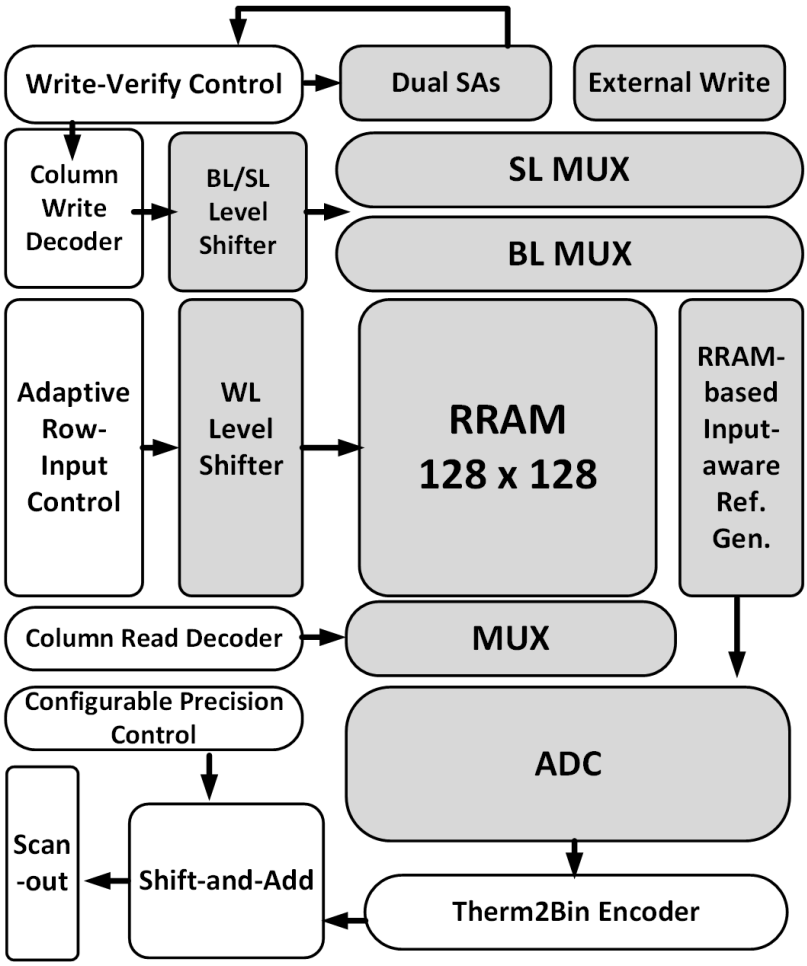


Figure 3.14: Top-level diagram of Flex-RRAM chip. The blocks in grey are custom-designed analog modules and the blocks in white are synthesized digital modules.

skipping the inputs of 0. The thermometer-to-binary encoder encodes the outputs of ADCs to 3-bit binary codes. The configurable precision controller and shift-and-add module work jointly to accumulate the partial sums with the corresponding significance.

3.4.2 Flex-RRAM Chip Design Features

In this section, we introduce the design features of Flex-RRAM in details. The main design features are listed as following: (1) configurable precision at run-time; (2) sparsity-aware adaptive row-input control; (3) RRAM-based input-aware reference generation; (4) On-chip write-verify control.

Using analog RRAM to efficiently represent multi-bit weight is a active research area,

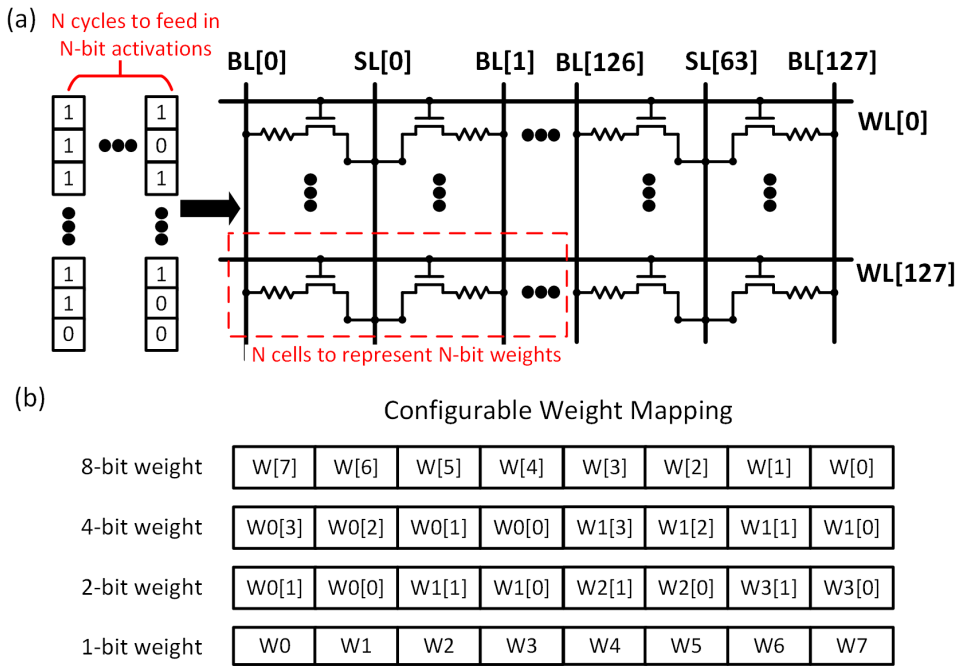


Figure 3.15: (a) The representation of N-bit weights and activations. Need N cycles to feed in N-bit activations, and N RRAM cells are grouped to represent N-bit weights. Shift-and-add modules are used at the periphery to handle the bit significance. (b) Configurable weight mapping scheme. Every 8 cells can be seen as one unit, where different portions of the 8 cells are grouped to represent the weights depending on the defined precision. For example, when weight precision is 4-bit, 8 cells are divided to two portions thus each weight consists of 4 RRAM cells.

however, the uniformity and reliability of the device remain as critical challenges. Thus, We still use industry-mature binary RRAM devices in this design where multiple cells are grouped to represent one weight. Figure 3.15(a) shows the representation scheme of N-bit activations and N-bit weights. N cycles are used to encode N-bit activations by asserting the input vector through WL bit by bit. N cells are grouped to represent N-bit weights accordingly. Shift-and-add module is used at the periphery to accumulate the partial sums with the corresponding significance. Figure 3.15(b) shows the configurable weight mapping scheme. Every 8 cells can be seen as one unit, where different portions of the 8 cells are grouped to represent the weights depending on the defined precision (1/2/4/8-bit). For example, when weight precision is 4-bit, 8 cells are divided to two portions thus each weight consists of 4 RRAM cells. Meanwhile, the shift-and-add module adjusts the

shift operation according to the significance of the corresponding weight column.

It is well known that the weights and activations typically show high sparsity [95], meaning that there are many zero weights and activations in DNN models, especially when the ReLU function is exploited as activation function. Thus, the throughput and energy-efficiency could be improved if we can effectively skip such zero values. In this design, we implement a sparsity-aware adaptive row-input controller to leverage the sparsity of input activations. We preprocess the input vector by counting the number of "1"s in the input vector and a threshold is set to trigger the assertion of the rows. During the preprocessing, we scan the input vector bit by bit, once the counting reaches the threshold value, the scanned rows will be asserted and the counter will be reset. Since the LRS of the TSMC RRAM cell is quite low, we set the threshold value to be 7 to make sure the column current won't become too high, meaning that there are only 7 "1"s applied to the rows at maximum. As an extreme case, if all the input bits are zero, the whole computation will be skipped. By exploiting the proposed scheme, we can effectively skip lots of unnecessary read operations and shift-and-add operations, thus improving the throughput and energy-efficiency. The simulation results will be shown in the following section.

As introduced in the section of XNOR-RRAM chip, we use external voltage supply to provide reference voltages for ADCs. In this chip, we design a input-aware reference generation module using RRAM array, which allows on-chip calibration by fine-tuning the resistance level of the RRAM cells. Figure 3.16 shows the configuration of RRAM-based input-aware reference array. H/L/U stands for HRS/LRS/Unformed-cell. If we define the potential outputs as 0-7, corresponding to 0-7 LRS cells, the first 7 rows with the fixed pattern duplicate the data patterns of 0-6, then the additional HRS cells (or unformed cells) in the tuning rows are activated to fine tune the reference columns to bias them in between every two data states. Depending on the relative ratio between LRS, HRS, and unformed cell, the reference levels can be tuned with different step sizes. The WL input to the fixed rows depends on the number of "1"s counted by the adaptive row-input controller, i.e., a

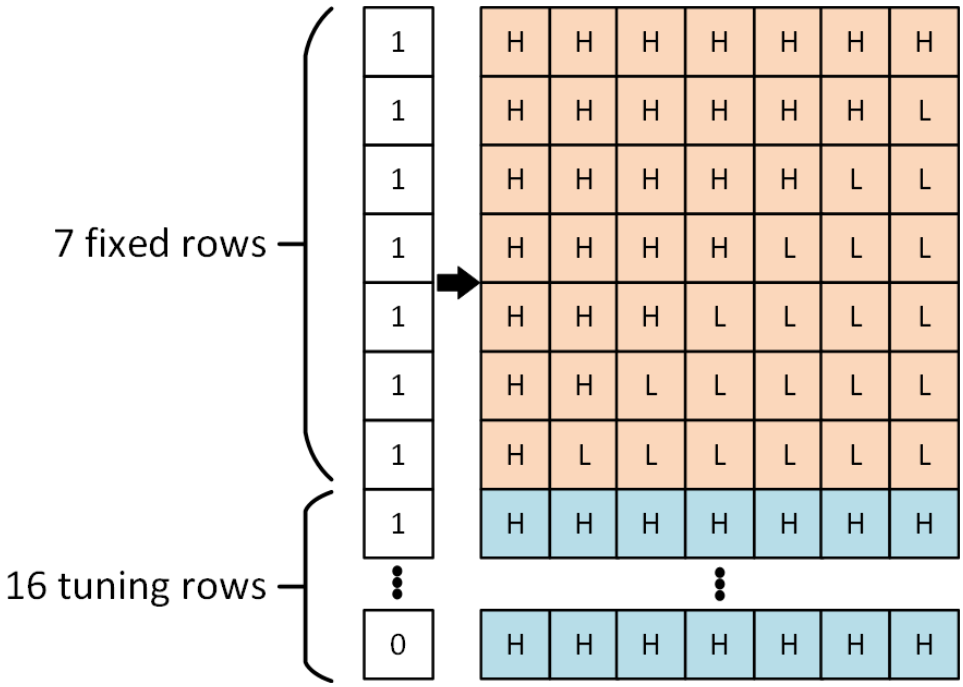


Figure 3.16: The configuration of RRAM-based input-aware reference array. H/L/U stands for HRS/LRS/Unformed-cell. If we define the potential outputs as 0-7, corresponding to 0-7 LRS cells, the first 7 rows with the fixed pattern duplicate the data patterns of 0-6, then the additional HRS cells (or unformed cells) in the tuning rows are activated to fine tune the reference columns to bias them in between two data states.

same number of fixed rows will be activated. For example, all 7 fixed rows will be activated in most cases as we set 7 as the threshold value that constrains the maximum number of activated rows for the data array as explained earlier. The WL input to the tuning rows is controlled by a 4-bit signal that determines the number of activated rows, i.e., from 0 to 15. With this, the reference columns could be fined tuned through closed-loop calibration. Since the patterning of reference array only needs to be performed once, the overhead of the calibration is negligible.

3.4.3 Flex-RRAM Simulation Results

Figure 3.17(a) presents the layout image of the whole chip that consists of 3 different macros. The Flex-RRAM macro is shown as highlighted. Figure 3.17(b) shows the detailed layout image of the Flex-RRAM macro. The main blocks include RRAM array, write

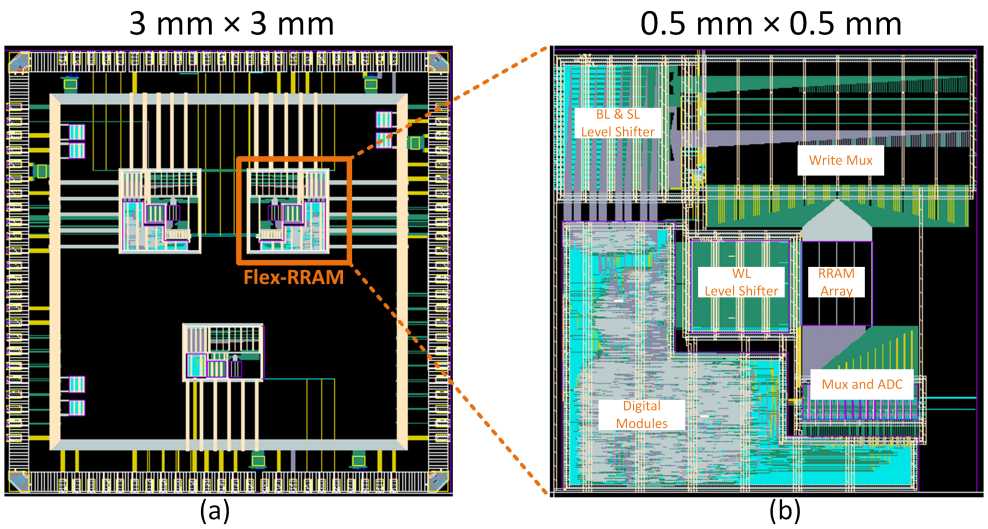


Figure 3.17: (a) Layout image of the whole chip which consists of 3 macros. The Flex-RRAM macro is highlighted. (b) Layout image of the Flex-RRAM macro. The main blocks include RRAM array, write MUXs, level shifters, ADCs, and digital controller.

MUXs, level shifters, ADCs, and digital controller. The total chip area is 3 mm by 3 mm while the Flex-RRAM macro takes around 0.5 mm by 0.5 mm. Since the chip is still in process at foundry, we only show results based on post-layout simulation.

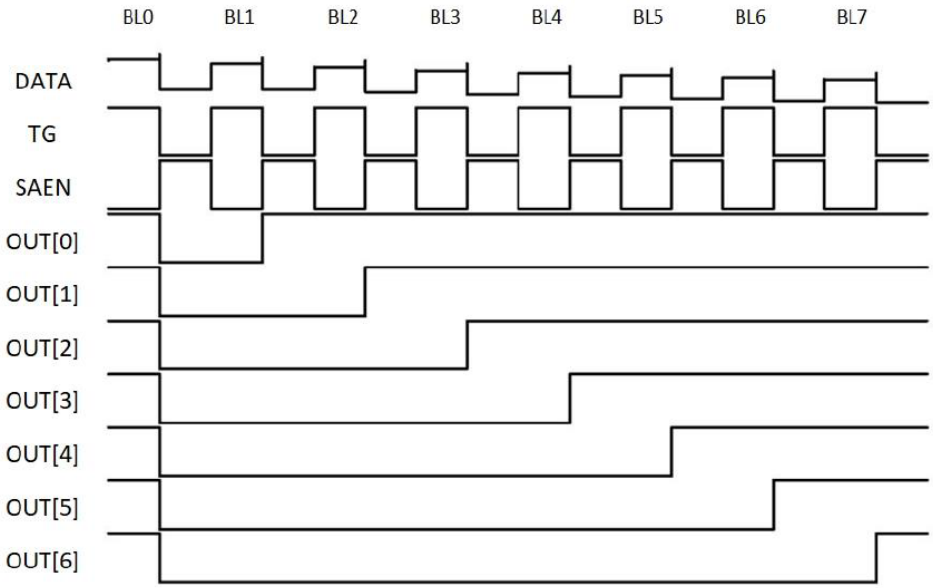


Figure 3.18: Simulated waveforms of sensing 7 bitlines configured to contain 0-7 LRS cells respectively. The correct 7-bit thermometer sensing results are well observed.

Figure 3.18 shows the waveforms of sensing 7 consecutive BLs in the same group that share one 3-bit ADC. The data pattern of the BLs are programmed to contain 0-7 "1"s respectively. It can be seen that the output results are correct, which will be converted to 3-bit binary code in the next through thermometer-to-binary encoder.

Table 3.3: Throughput and Energy-efficiency under Different Sparsity Factor

Input vector	Throughput (GOPS)	Energy-efficiency (TOPS/W)
5% on	25.0	45.15
50% on	14.1	15.28
100% on	9.0	7.71

To analyze the effectiveness of the adaptive input-sparsity control, we set up 3 cases to simulate the throughput and energy-efficiency of the design, where 5%, 50%, and 100% of input bits are "1" respectively. The precision is configured to be 1-bit weight and activation. Table 3.3 summarizes the simulation results in different cases. It can be seen that both the throughput and energy-efficiency could be improved by $\sim$3X when the input sparsity increases from 0 to 95%. For the average case where the input sparsity is 50%, the throughput is 14.1 GOPS and the energy-efficiency could achieve 15.28 TOPS/W. It can be seen that the energy-efficiency of Flex-RRAM chip when configured for binary neural networks is less than XNOR-RRAM chip, which can be explained by two main reasons. First, the resistance of TSMC RRAM device is less than that of Winbond RRAM device, leading to larger column current during read-out operation. Higher cell resistance values are preferred to achieving higher energy-efficiency. Second, XNOR-RRAM chip only contains RRAM array and ADCs for demonstrating the CIM operation, the accumulation happens in software after reading out the data from the chip. Flex-RRAM chip includes more digital modules such as shift-and-add units, accumulation units, and the related control units, which induce additional energy consumption.

3.5 Summary

This chapter presents two generations of RRAM-based testchips taped-out using commercial RRAM process, namely XNOR-RRAM (Winbond 90 nm process) and Flex-RRAM (TSMC 40nm process) that demonstrate the feasibility of CIM operation for accelerating the inference of DNNs. XNOR-RRAM chip is dedicated for binary neural networks where weights and activations are quantized to binary state, and Flex-RRAM chip can support 1bit-to-8bit flexible precision configurability at run-time. Flex-RRAM also features adaptive input sparsity control which takes advantage of the high sparsity nature of typical DNN models to improve the throughput and energy-efficiency. In addition, Flex-RRAM also supports on-chip input-aware reference generation that provides fine-grained tunability of ADC reference voltages, and on-chip write-verify control that can speed up the weight programming phase drastically. The measurement result of XNOR-RRAM chip shows that it can achieve energy-efficiency of 24.1 TOPS/W at 1.2V while maintaining a acceptable classification accuracy on CIFAR-10 dataset. As the Flex-RRAM chip is still in fabrication at foundry, the post-layout simulation result shows that it could achieve 15.28 TOPS/W (when configured for BNN) at 0.9V assuming the sparsity of input activations is 50%.

CHAPTER 4

IMPACT OF DEVICE NONIDEALITIES ON TRAINING AND INFERENCE

4.1 Introduction

A major challenge for designing eNVM-based CIM accelerator is the device nonidealities that prevent the system from achieving software-equivalent accuracy. Even though using binary eNVM devices with large ON/OFF ratio is viable to implement arbitrary-precision weights while showing better resiliency to device nonidealities, it is more attractive to utilize the multi-level per cell (MLC) capability of the eNVM devices which can further improve the area- and energy-efficiency. For in-situ training, which involves a substantial amount of weight update, the MLC eNVM devices typically suffer from various non-ideal characteristics including nonlinear and asymmetric conductance tuning behavior, conductance variation, device-to-device variation (D2D), cycle-to-cycle variation (C2C), and write endurance. For inference, write-verify technique can be used to enhance the quality of programming before deployment. However, analog eNVM devices typically suffer from the retention problem, i.e., the resistance drift, which results in errors of weight values that cause inference accuracy degradation. In this chapter, we conduct a comprehensive investigation through the training and inference of a representative CNN with CIFAR-10 dataset. A PyTorch-based simulation framework is developed to incorporate the aforementioned device nonidealities into the CNN model.

We perform the analysis with an virtual 8-bit device with assumed properties exhibited by typical eNVM devices. Our simulation results suggest that: (1) the training accuracy is more sensitive to the asymmetry of conductance tuning than the nonlinearity, the high nonlinearity can be tolerated if the potentiation (P) and depression (D) have the same polarity while asymmetric P/D (which is typically the case in today's devices) significantly

degrades the training accuracy; 2) the conductance range variation does not degrade the training accuracy significantly, instead, a small variation can even reduce the accuracy loss introduced by asymmetry; 3) D2D variations can also remedy the accuracy loss due to asymmetry while C2C variations lead to dramatic accuracy degradation; 4) The accuracy degradation will not be noticeable if the endurance cycles (defined as the number of programming cycles that cause the conductance tunable dynamic range decays by 50% are more than 7,000 cycles; 5) Different conductance drifting modes affect the inference accuracy differently, and the best case is where the conductance is drifting up/down randomly.

4.2 Evaluation Framework Setup

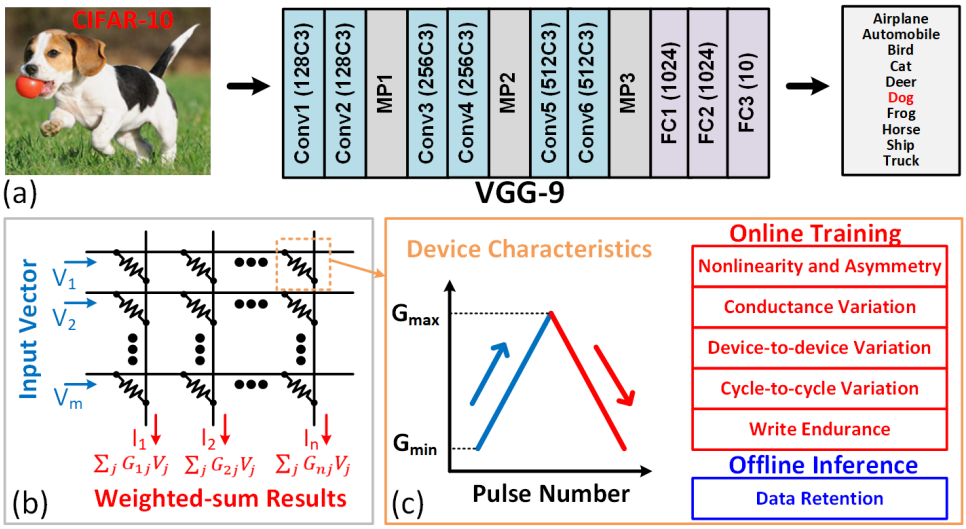


Figure 4.1: (a) The CNN model used for CIFAR-10 dataset (a variant of VGG-Net), consisting of 6 convolutional layers (CONV), 3 maxpooling layers (MP) and 3 fully connected layers (FC). (b) Weighted-sum operation in an eNVM based resistive synaptic crossbar array structure where input vectors are encoded into read voltages and weighted-sum results are obtained in terms of column current. (c) The value of weights are updated by tuning the conductance of synaptic devices by applying positive and negative pulses. Ideally, the conductance is being linearly tuned without any variations or endurance issues. However, the device nonidealities will introduce errors thus degrading the accuracy.

In a resistive synaptic array, the weight matrices are represented by the conductance matrices of the eNVM devices as shown in Figure 4.1(b). The crossbar array consists

of perpendicular rows and columns with the resistive synaptic device sandwiched at each cross-point. The weight values are mapped to the conductance of the devices. It is worth noting that the conductance can only represent positive values (0, 1] while the weights in algorithm typically are in range [-1, +1]. To enable the mapping of conductance to negative weights, a dummy column can be used to perform the conversion as shown below:

$$W = 2 \times G - J \tag{4.1}$$

where J is the matrix of all ones with the same dimension as weight matrix W. Thus -1 is mapped to the minimum conductance and +1 is mapped to the maximum conductance. Then the weighted-sum operation is performed in a parallel fashion at analog domain: the input vectors are encoded into read voltages applied to all the rows, the weighted sum results are obtained in terms of the summed currents at the end of each column. Typically, there are neuron circuits placed at the periphery to convert the analog current to digital format output for further communications [96]. The weights are updated by tuning the conductance state of the synaptic devices by applying positive and negative pulses as depicted in Figure 4.1(c). Ideally, the conductance of the device is expected to be perfectly tuned, i.e., by linear and symmetric step, without variations or endurance degradation. However, the reported eNVM devices typically suffer from various non-ideal effects, among which the nonlinearity and asymmetry of conductance tuning, device variations, and write endurance will affect the in-situ training process.

In this work, we build up a VGG-like CNN model, namely VGG-9 (Figure 4.1(a), inspired from [86]), to be evaluated on CIFAR-10 dataset. VGG-9 consists of six convolutional layers and three fully connected layers. ReLU is used as the activation function. Every two convolutional layers are followed by one max pooling layer to sub-sample the feature size. The dimension of the input image is $3 \times 32 \times 32$, and the kernel size is 3×3. Since the number of multi-level conductance states in one synaptic device is limited, the weight precision has to be constrained by fixed-point quantization in the algorithm. The

prior work [91] has shown near-software inference accuracy on the binary neural network, however it is known that the weight precision requirement of online training is higher than that of inference. To analyze the impact of weight precision on the training accuracy, we adopt a methodology inspired from [97] to quantize the weight and the gradient during training, where the continuous values are quantized to k-bit with the uniform step θ:

$$\theta(k) = 2^{1-k}, k \in Z_+ \tag{4.2}$$

Then the quantization function can be formulated as the following:

$$(x, k) = clip\{\theta(k) \times round(\frac{x}{\theta(k)}), -1 + \theta(k), 1 - \theta(k)\} \tag{4.3}$$

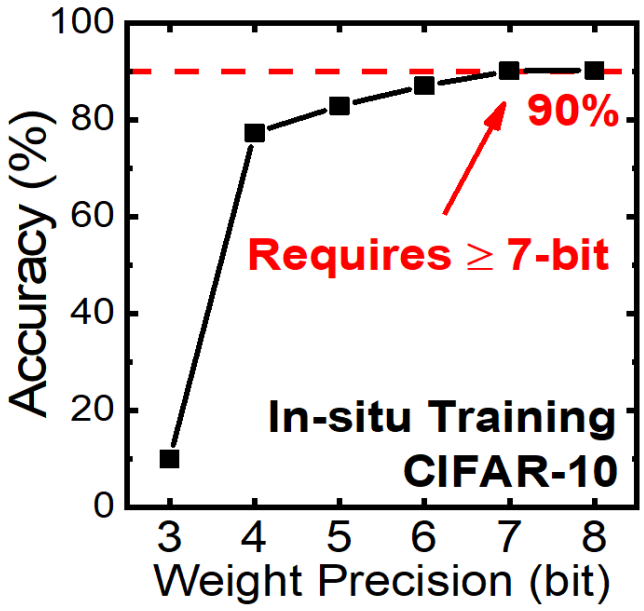


Figure 4.2: Training accuracy on CIFAR-10 dataset with different weight precisions while activation precision is fixed at 8-bit. 8-bit weight precision is required to achieve near-optimal accuracy.

Figure 4.2 shows the saturated training accuracy (averaged from last 10 training epochs) on CIFAR-10 dataset with different weight precisions while the activation precision is fixed

at 8-bit. The model is trained for 200 epochs with mini-batch size of 128, the learning rate is set to 1 and scaled to 0.25 at epoch 100 and 0.125 at epoch 150 respectively. The result suggests that 8-bit is required to achieve near-optimal accuracy (90%). Compared to the prior results on MNIST dataset [98], the weight precision requirement increases from 6-bit to 8-bit for the more complex CIFAR-10 dataset. In the following simulations, we fix the weight precision as 8-bit, and modify the vanilla stochastic gradient descent (SGD) algorithm to incorporate the non-ideal device characteristics in both training and inference of the CNN.

Figure 4.3 shows the pseudo-code of the functions for each non-ideal effect. We use weight W instead of conductance G in the expressions for simplicity. In this paper, W and G or (ΔW and ΔG) are interchangeable. For the online training, each weight has its conductance tuning function determined by the corresponding properties of nonlinearity/asymmetry expressed as in equation (1). Ideally, the minimum weight W_{min} equals -1, the maximum weight W_{max} equals 1, the index number of conductance state S_{index} is in the range $[1, 256]$, and the nonlinearity factor A equals 0. Taking the current weight value (W_0) and the ideal weight change (ΔW) as inputs, the actual updated weight (W_{actual}) can be calculated through equation (2-3).

As illustrated in Figure 4.4 (P/D nonlinearity factor is 5/-5 as an example), for the transition from G_i to G_{i+1}, where ΔG_1 equals +30 steps, the current state index S_P of G_i is solved from the inverse function of the P tuning curve. Next, S_P is updated to $S_P{}'$ by adding ΔG_1. Thus the new conductance ΔG_{i+1} can be solved on the P curve. For the transition from ΔG_{i+1} to ΔG_{i+2} where ΔG_2 equals -30 steps, S_D solved from the inverse function of the D curve is used as the current state index instead of $S_P{}'$, then $S_D{}'$ and the new conductance ΔG_{i+2} are solved through the D curve. Ideally ΔG_{i+2} should equal to G_i, however due to the P/D asymmetry, a deviation error is introduced. It is noted that S_P/SD ($S_P{}'$/$S_D{}'$) may be a decimal value due to the mismatch between the conductance values on P/D curve while the number of update steps will always be an integer as ΔG

Modified SGD Algorithm: Incorporating Non-ideal Characteristics

Require $W_0, \Delta W$ from normal SGD

Procedure *Nonlinearity and Asymmetry*

$W = f(W_{min}, W_{max}, S_{index}, A)$ ← Nonlinear mapping function (1)

S_{index} *is the index of the conductance state.*

A *is the nonlinearity controlling factor.*

$S_{actual} = f^{-1}(W_0) + Normalized(\Delta W)$ (2)

$W_{actual} = f(W_{min}, W_{max}, S_{actual}, A)$ (3)

Procedure *Conductance Variation*

Add variation to the maximum conductance state

$W = f(W_{min}, W_{max} + N(\sigma), S_{index}, A)$ (4)

Procedure *Device-to-device Variation*

Add variation to the nonlinearity factor of each weight

$W = f(W_{min}, W_{max}, S_{index}, A + N(\sigma))$ (5)

Procedure *Cycle-to-cycle Variation*

Add variation to the weight change at every update cycle

$W = f(W_{min}, W_{max}, S_{index}, A) + N(\sigma)$ (6)

Procedure *Write Endurance*

Scale the weight change at every update cycle

for each weight, initialize $counter = 0$

for each update iteration:

$counter = counter + N$,

N *is the equivalent number of pulses applied* based on ΔW

$\Delta W_{actual} = \Delta W\ (1 - \mathrm{r})^{counter}$ (7)

r *is the reduction ratio, e.g.,* 0.001

Procedure *Data Retention*

$W_{inference} = W_{trained}(\frac{t}{t_0})^{v}$ (8)

v *is the drift coefficient that controls the drifting rate.*

t *is the retention time,* t_0 *is the time constant.*

Figure 4.3: The pseudo-code of the functions used to incorporate the non-ideal device characteristics during training. The current weight W_0 and the ideal weight change ΔW from the normal SGD algorithm are used as inputs to generate the actual weight. $N(\sigma)$ is the variation in normal distribution with standard deviation of σ.

is quantized to 8-bit. This means from algorithm perspective, weight value is equivalently in floating-point precision but the gradient (i.e., ΔW) is quantized to 8-bit. For different variations, we add randomness in normal distribution to different parameters accordingly as show in equation (4-6). To take write endurance into consideration, we add a variable

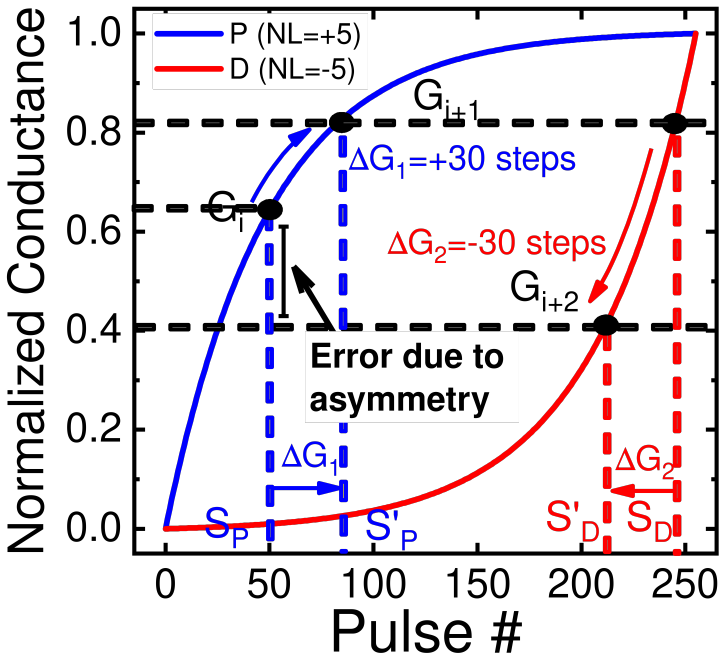


Figure 4.4: The conductance update procedure with nonlinearity. The nonlinearity factor of potentiation (P) and depression (D) is set to $5/-5$ as an example. For the transition from G_i to G_{i+1}, the current state index S_P of G_i is solved from the inverse function of the P tuning curve. Next, S_P is updated to $S_P{}'$ by adding ΔG_1. Thus the new conductance G_{i+1} can be solved on the P curve. For the transition from G_{i+1} to G_{i+2}, S_D solved from the inverse function of the D curve is used as the current state index instead $S_P{}'$, then $S_D{}'$ and G_{i+2} are solved through the D curve. A deviation error is introduced due to the P/D asymmetry.

as counter to keep track of the accumulative number of pulses that has been applied to each weight according to the amount of actual weight change and scale the weight change accordingly as expressed in equation (7). For the inference, the model is pre-trained by pure software and we assume that all the weights can be tuned to the ideal value through write-verify technique. Therefore, we only consider the impact of data retention on the inference accuracy. The drifting behavior is modeled by equation (8) where a drift coefficient v is used to control the drifting rate. It should be noted that our developed framework is flexible to exploit any variation/error modeling functions, and scalable to be applied to any arbitrary mainstream neural network structures including MLP, CNN, and recurrent neural network (RNN).

4.3 Impact of Device Nonidealites on Training and Inference

4.3.1 Nonlinearity and Asymmetry

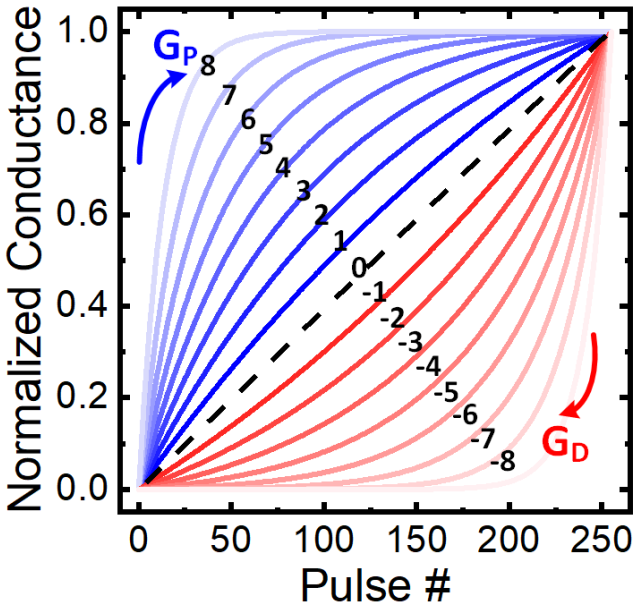


Figure 4.5: . Different nonlinearities of the potentiation and depression tuning curves from the fitted model in [98]. The nonlinearity is labeled from 8 to -8.

The linearity of conductance tuning refers to the linearity of the curve between the conductance and the number of programming pulses, which is desired to be linear and symmetric to perfectly model the weight change in algorithm. However, the conductance of the realistic synaptic devices typically does not vary linearly with the number of applied pulses. Moreover, the trajectory of the weight increasing process differs from that of the weight decreasing. This nonlinearity/asymmetry is undesired because it will deviate the weight change from its designated value. It is worth noting that the nonlinearity/asymmetry only affects the online training process as the conductance needs to be continuously tuned, while for offline inference, the conductance can be nearly perfectly tuned through iterative programming with write-verify technique [99]. To quantitatively analyze its impact on the training accuracy, we adopt the extracted nonlinearity behavioral model in [98]. Figure 4.5

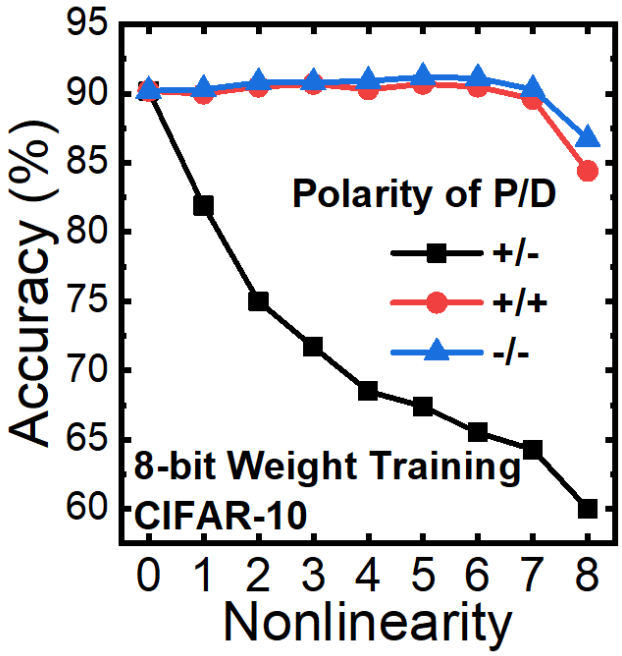


Figure 4.6: The training accuracy with different polarities of nonlinearity during potentiation and depression. High nonlinearity (up to 6) is tolerable when P/D has the same polarity, otherwise the accuracy degrades dramatically with the increasing asymmetric nonlinearity.

shows several example conductance tuning curves with different nonlinearities (labeled as 8 to -8) by adjusting the value of A. The black dotted line in the middle is the ideal case where the conductance tuning is linear and symmetric. When the potentiation (P) and depression (D) has the nonlinearity with both the same magnitude and polarity, the weight update will be nonlinear but symmetric, when potentiation and depression process has the nonlinearity with different polarities, the weight update will be nonlinear and asymmetric. We simulate 3 scenarios where the polarity of P/D is positive/negative, positive/positive, and negative/negative respectively. As shown in Figure 4.6, the training accuracy remains as 90% for blue and red lines even when the nonlinearity magnitude is 6, that means high nonlinearity can be well tolerated if P/D has the same polarity, i.e., good symmetry. However, for the common situation with asymmetric P/D, the training accuracy degrades dramatically with the increasing nonlinearity. With a moderate nonlinearity magnitude of 1, the training accuracy already degrades to 82%. Therefore, symmetry plays a more critical

role than linearity in maintaining a good training accuracy. There are a few strategies to address the nonlinearity/asymmetry issue [100]. For example, the utilization of non-identical pulses for programming could improve the linearity of the TaO_x/TiO_2 device as reported in [101]. However, the non-identical pulse generation introduces additional overheads in terms of peripheral circuits because the amplitudes or the duration of the pulses require calibration by reading out the current conductance state before the programming operation. On the other hand, several capacitor-assisted weight-cell designs that separate the different significance bits (e.g., 3T1C+2PCM [47], 2T-1FeFET [49]) have been proposed, which demonstrate much improved linearity and symmetry, but at expense of additional overhead on area and power.

4.3.2 Conductance Range Variation

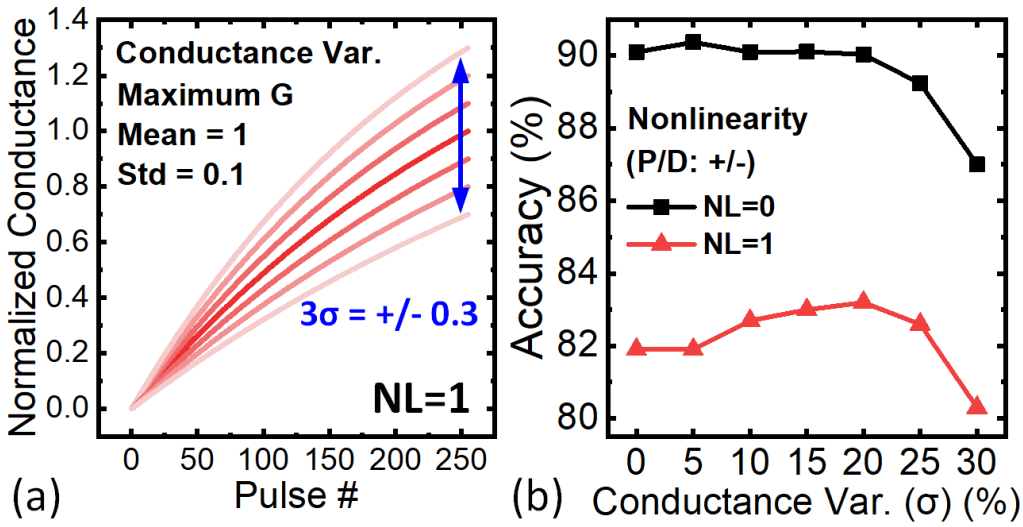


Figure 4.7: (a) Illustration of the conductance range variation under nonlinearity (NL) of 1. Variations are added to the maximum conductance state with standard deviation (σ) in terms of percentage. Here $\sigma = 10\%$, thus $3\sigma = 30\%$ is shown as an example. (b) The training accuracy with conductance variation under different P/D nonlinearities ($NL = 0/0$ and $+1/-1$). A small variation ($< 20\%$) does not degrade the accuracy, instead, it remedies the accuracy loss introduced by the asymmetric nonlinearity. Noticeable degradation occurs when variation exceeds 20%.

The variability in eNVMs is a major issue in conventional memory application. In con-

trast, the neural networks are typically less sensitive to the variations thanks to the intrinsic excessive weight connections and the iterative weight update during training. However, the degree of the network's resiliency to variations highly depends on the specific algorithm, network structure, and the complexity of the target task.

In this work, to analyze the impact of conductance range variation on the training accuracy, we add the variation in terms of the percentage to the maximum conductance state as it changes the conductance range most. Figure 4.7(a) shows several P curves (the nonlinearity is +1) with the variation on $G_max(\sigma = 10\%)$ as an example. We sweep the value of from 5% to 30% for two cases where the nonlinearity is 0 and 1 respectively. The simulated results in Figure 4.7(b) indicates that a small variation ($< 20\%$) does not degrade the training accuracy, conversely, it remedies the accuracy loss introduced by the asymmetric nonlinearity. This could be explained by the fact that small random disturbance may compensate the deviation introduced by the asymmetry. A noticeable degradation occurs when the variation exceeds 20%.

4.3.3 Device-to-Device Variation

The effect of device-to-device variation can be analyzed by adding the variation to the nonlinearity baseline of each synaptic device. Figure 4.8(a) illustrates the case of several P curves where the baseline nonlinearity equals 1 and the standard deviation equals 0.5. Similarly, we investigates two cases where the baseline nonlinearity magnitude is 0 and 1 respectively. As shown in Figure 4.8(b), for the case with ideal baseline, the training accuracy continues degrading with the increasing device-to-device variation as expected because the tuning behavior of many devices become asymmetric due to the random variation. In contrast, for the common case that has a moderate nonlinearity of +1/-1, device-to-device variation improves the accuracy from 82% to 86%. We speculate the reason for the accuracy recovery could be that the neural network can adapt itself to rely more on those devices bearing less nonlinearity and asymmetry.

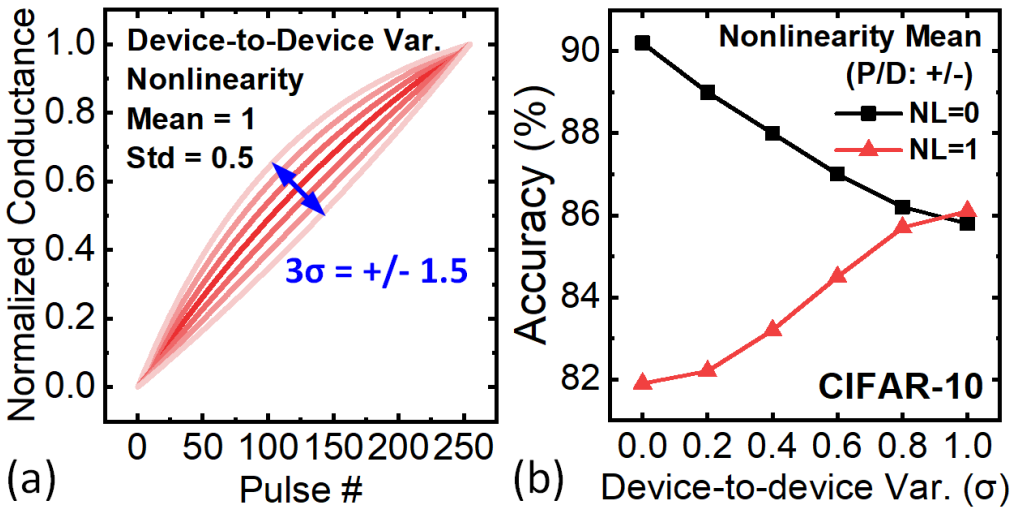


Figure 4.8: (a) Illustration of the device-to-device variation where $\sigma = 0.5$. D2D variation is incorporated by adding the variation to the nonlinearity baseline of each synaptic device. Here $\sigma = 0.5$, thus $3\sigma = 1.5$ is shown as an example. (b) The training accuracy with device-to-device variation under different baseline P/D nonlinearities ($NL = 0/0$ and $+1/-1$). For the common case that has a moderate nonlinearity of +1/-1, D2D variation improves the accuracy because the neural network could adapt itself to rely more on the devices that have more symmetric conductance tuning behavior.

4.3.4 Cycle-to-Cycle Variation

The impact of cycle-to-cycle variation on the training can be analyzed by introducing a random error to the conductance change when each tuning pulse is applied during weight update as illustrated in Figure 4.9(a), the amount of the standard deviation is expressed in terms of the percentage of the entire weight range. The result in Figure 4.9(b) shows that a cycle-to-cycle variation of 5% significantly degrades the accuracy to be less than 40%. It should be noted that the degree of cycle-to-cycle variation's impact on the training accuracy strongly depends on the weight precision, i.e., the number of states in the synaptic device. Here based on the assumption of 256 states within one device, even 1% variation means a difference of $\sim$2.5 states, resulting in a severe disturbance to the weight update. It is expected that a lower weight precision will suffer less from cycle-to-cycle variation due to the reduced relative disturbance to the sparser conductance states. This agrees with the results in [98] where the weight precision is 6-bit, the accuracy does not degrade too much

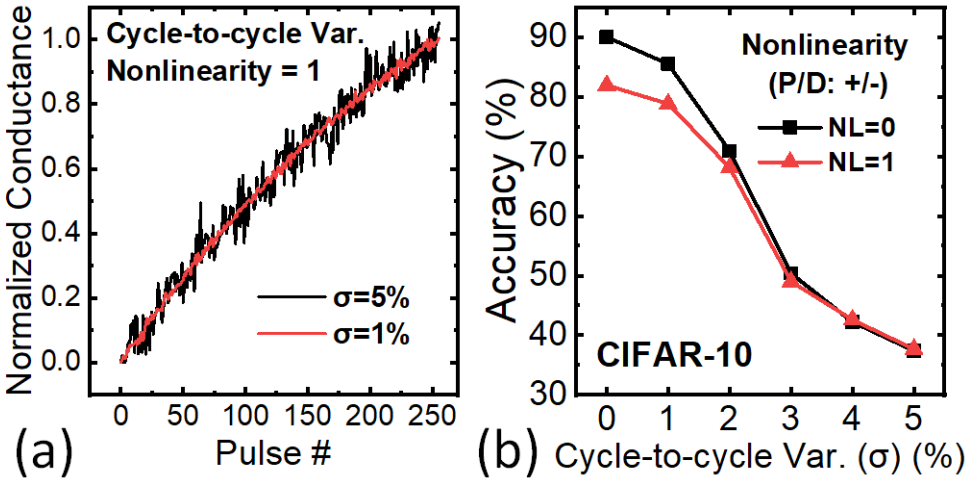


Figure 4.9: (a) Illustration of the cycle-to-cycle variation with $\sigma = 5\%$ and $\sigma = 1\%$ respectively. The amount of σ is in terms of the entire conductance range. C2C variation is incorporated by adding the variation to the conductance change at every programming pulse. (b) The training accuracy with C2C variation under different baseline P/D nonlinearities ($NL = 0/0$ and $+1/-1$), showing continuous degradation with the increasing variation. Accuracy becomes less than 40% when σ is 5%.

until σ becomes larger than 2%.

4.3.5 Write Endurance

In memory applications, the write endurance is referred to as the number of times that a memory cell can be programmed before the write failure occurs. Binary eNVM devices typically can achieve ¿106 endurance cycles of switching between the high resistance state and the low resistance state. However, for the analog eNVM devices, the definition of the endurance cycle should be different as the conductance is being incrementally tuned at every write pulse. As characterized in [103], the analog switching performance degrades as update number increases since the conductance change (ΔG) cannot remain as a constant value throughout the entire tuning process. As depicted in Figure 4.10(a), the effective ΔG will decrease over the programming pulses, and the conductance will eventually not be able to be tuned anymore. To quantitatively study the endurance effect, we adopt the endurance

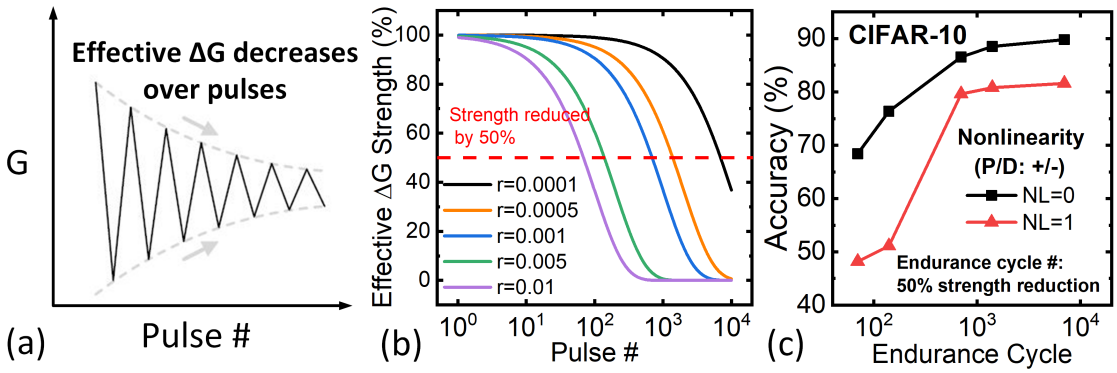


Figure 4.10: (a) Endurance degradation during weight update. The effective ΔG decreases over programming pulses [102]. (b) Quantitative illustration of the endurance degradation. With different reduction ratio, the ΔG strength decays at different rate and the conductance will be eventually unchangeable (write failure) after a certain number of programming cycles. We define the endurance cycle to be the number of cycles that cause the ΔG strength decays by 50%. (c) The training accuracy on CIFAR-10 with different endurance cycles. A good endurance property ($> 7,000$ cycles) is critical for the network to avoid noticeable accuracy degradation.

behavior model in [102], assuming the strength of ΔG decreases over write pulses, which is expressed as the following equation, where ΔG_0 is the expected conductance change without endurance effect, r is the reduction ratio that controls the rate of the ΔG strength degradation, and N_{PULSE} is the cumulative number of write pulses applied to the device.

$$\Delta G = \Delta G_0 (1 - r)^{N_P ULSE} \tag{4.4}$$

As shown in Figure 4.10(b), with different reduction ratios, the ΔG strength decays at different rate and the conductance will be eventually unchangeable (write failure) after a certain number of programming cycles. In this work, we define the endurance cycle to be the number of cycles that cause the ΔG strength decays by 50%, thus the endurance cycle equals 70, 140, 700, 1,400, and 7,000 respectively for r swept from 0.01 to 0.0001. In the simulation, we keep track of the equivalent number of pulses applied to each weight and downscale the actual weight change accordingly. Figure 4.10(c) shows the training accuracy on CIFAR-10 with different endurance cycles. A similar trend is observed for both cases with asymmetry and nonlinearity of 0/0 and +1/-1. A good endurance property

($> 7,000$ cycles) is critical for the network to avoid noticeable accuracy degradation for CIFAR-10 dataset, while the endurance requirement for MNIST dataset is only $\sim$700 cycles [102]. It is noted that the requirement on the endurance cycle is highly task-dependent, i.e., the number of weight update iterations needed for convergence.

4.3.6 Resistance Drift

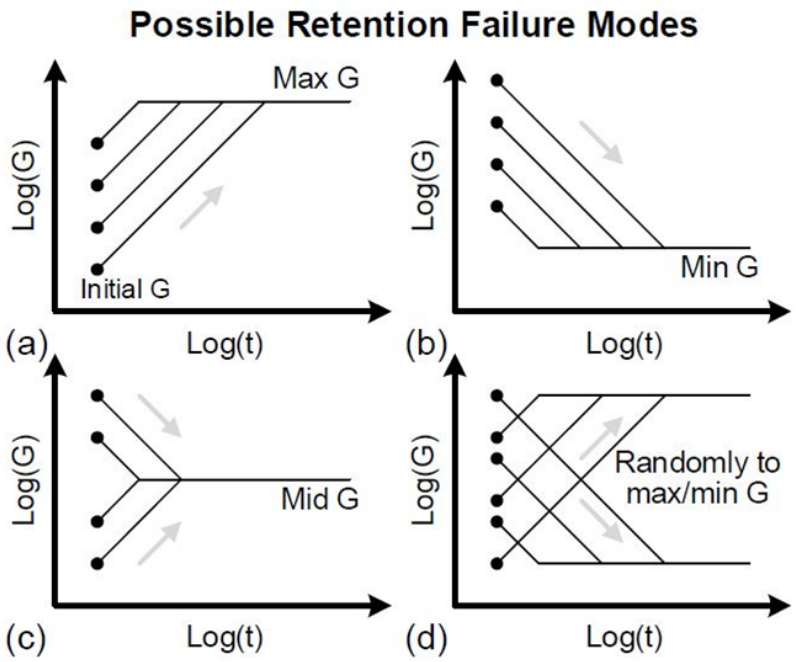


Figure 4.11: Illustration of different possible retention failure modes [102]. The conductance drifts towards (a) maximum conductance state, (b) minimum conductance state, (c) intermediate conductance state, (d) maximum/minimum conductance state with randomness.

Data retention refers to the capability of the eNVM devices to maintain its programmed state over a certain period of time. In memory applications, the typical requirement on retention time is > 10 years at $85°\mathrm{C}$ While many binary eNVM devices have shown qualified property regarding to this requirement, there are no reported data for analog eNVM showing such retention due to the instability of intermediate conductance states [104]. In this work, we generalize the analysis by considering 4 different drift modes as shown in Figure 4.11. The conductance can either drift toward its (a) maximum, (b) minimum, or

(c) intermediate conductance states, which have all been reported in the retention measurement of binary eNVMs [105, 106]. In addition, we also include the case (d) where the conductance may randomly drift to maximum/minimum conductance state. Since weights are frequently updated during online training, the impact of the long-term retention effect is negligible. We assume that the network is pre-trained and all the weights are perfectly programmed to the desired conductance state through write-verify programming protocol. Then to analyze its impact on the inference accuracy, we adopt the drifting behavior model that is widely used in PCM [107], which is shown as the following:

$$G = G_0(\frac{t}{t_0})^v \tag{4.5}$$

where G_0 is the initial conductance, t is the retention time, v is the drift coefficient and t_0 is the time constant which is assumed to be 1 second in this work.

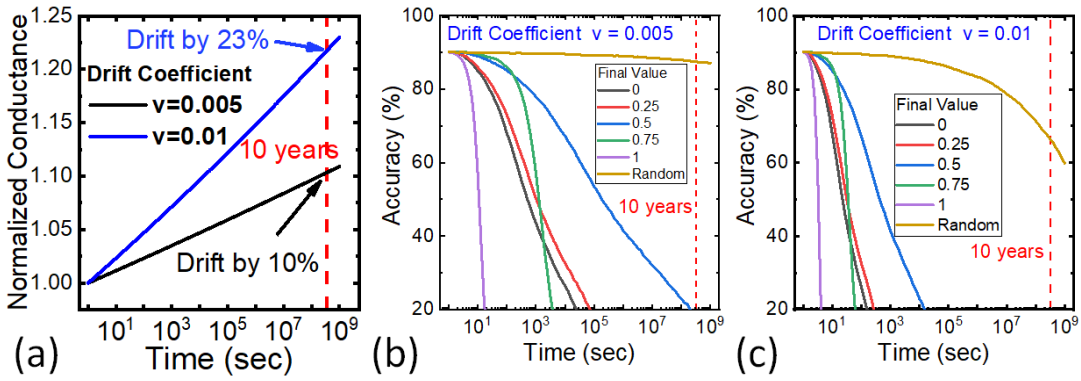


Figure 4.12: (a) Illustration of the drifting effect with different drift coefficients. By 10 years, the conductance is estimated to drift by $\sim 10\%$ and $\sim 23\%$ with the drift coefficient of 0.005 and 0.01 respectively. (b) The inference accuracy as a function of retention time with different failure modes at drift coefficient of 0.005. (c) The inference accuracy as a function of retention time with different failure modes at drift coefficient of 0.01. The normalized conductance is assumed to drift towards 0, 0.25, 0.5, 0.75, and 1 respectively. Different retention modes affect the inference accuracy differently and the best case is drifting towards maximum/minimum with randomness.

Figure 4.12(a) shows the example of the conductance drifting effect with different drift coefficients. By 10 years, the conductance is estimated to drift by $\sim 10\%$ and $\sim 23\%$ with

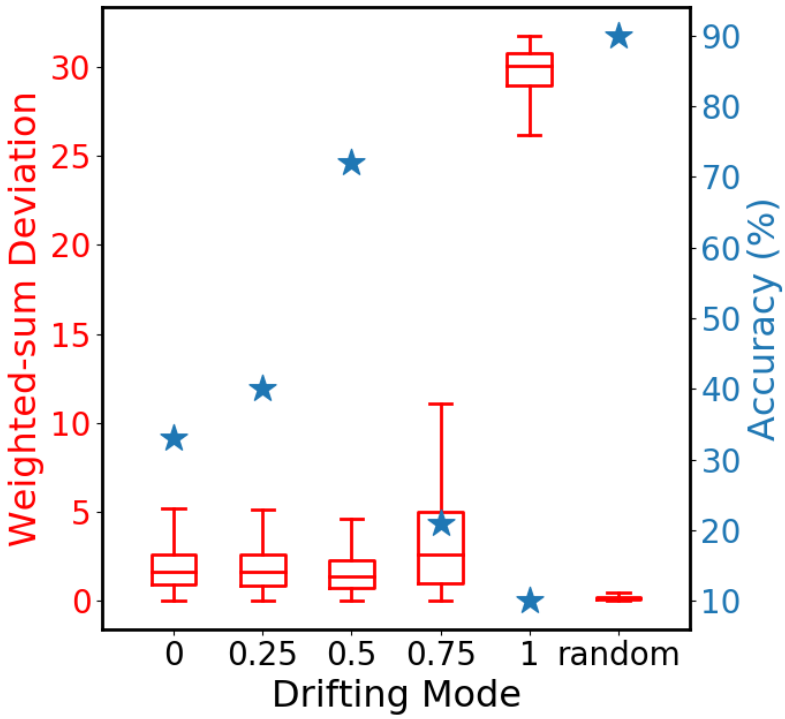


Figure 4.13: The magnitudes of the weighted-sum deviation (of the last convolutional layer as an example) and the inference accuracy for different drifting modes sampled at $t = 3600s$ with drifting factor $v = 0.005$, showing an inverse correlation between the magnitudes and the accuracy. The random drifting case retains the best inference accuracy with a smallest weighted-sum deviation.

the drift coefficient of 0.005 and 0.01 respectively. Figure 4.12(b-c) show the degradation of the inference accuracy with different drift modes over retention time at drift coefficient of 0.005 and 0.01 respectively. The results suggest that drifting to intermediate states leads to less accuracy degradation than drifting to maximum/minimum state. Among the drifting modes with determined final state, the case with the final state of 0.5 achieves the best accuracy. Compared to the scenarios with determined final state, random drifting shows much less accuracy degradation, which can retain 87% at 10 years with drift coefficient of 0.005. When v is increased to 0.01, the final accuracy at 10 years degrades to 67%. The trends

agree with the observations on MNIST dataset in [102]. The difference between network's resiliency to different drifting modes can be attributed to the difference of the weighted-sum deviations over retention time. Figure 4.13 shows the magnitude of the weighted-sum deviation for the last convolutional layer as an example, the data is sampled at retention time of one hour with the drifting factor of 0.005. For the case of random drifting, the randomness helps the compensation between the increasing weights and decreasing weights, thus the final weighted-sum results are much less deviated from the ideal value, which leads to the best inference accuracy as depicted by the star. In principle, the drifted conductance states in the inference engine can be refreshed by write operation occasionally at the expense of additional calibration steps.

To address the challenge posed by resistance drift, one potential approach is to reduce the drift coefficient through device engineering. As reported in [108], IBM's projected-PCM can achieve a drift coefficient of 0.002, which is 50X smaller than the typical value. However, this is achieved at the expense of limited conductance range. Another approach is to compensate from algorithm perspective. In [109], IBM proposes a so-called slope correction technique to compensate the drift effect, which inserts a time dependent correction term to be multiplied with the weighted sum results before activation to perfectly cancel the deviation caused by the reduction of conductance. However, this technique incurs significant overheads not only for storing the correction term parameters for each memory cell, but also for the additional exponential computation. Overall, we think there is currently not a recognized effective and practical method to overcome the challenge caused by the drift effect while it is a very critical issue to maintain a high inference accuracy.

4.3.7 Benchmark with Realistic eNVM Devices

In the recent years, many resistive synaptic devices have been reported with various characteristics. In this section, we select several representative eNVM devices reported in literatures: TaO_x/HfO_x [44], EpiRAM [110], HZO FeFET [45], and benchmark their perfor-

Table 4.1: Benchmark of Realistic Synaptic Devices

Device Type	Virtual Device	TaO_x/HfO_x [44]	EpiRAM [110]	HZO FeFET [45]
# of conductance states	256	128	64	32
Weight precision	8-bit	7-bit	6-bit	5-bit
Nonlinearity (LTP/LTD)	1/-1	0.04/-0.63	0.5/-0.5	1.75/1.46
D2D variation (σ)	0.1	0	0	0
C2C variation (σ)	1%	3.7%	2%	0.5%
Endurance (cycles)	7,000	∞	∞	∞
Accuracy on CIFAR-10	77.6%(90.2%)	78.7%(90.1%)	84.5%(87.0%)	81.2%(82.9%)

mance on training accuracy with CIFAR-10 dataset. In addition, we add a virtual device that combines the aforementioned non-ideal characteristics with moderate assumptions. The characteristics of those devices and the corresponding simulated training accuracy are summarized in Table 4.1. Please note that the baseline training accuracy is different for different devices due to different number of multi-level states. For example, with 32 states (i.e., 5-bit weights) demonstrated in [45], the baseline accuracy can only achieve 82.9% in the ideal case. The unreported characteristics of the practical devices are assumed to be ideal, e.g., we assume no D2D variation and infinite endurance cycles for the cited realistic devices. The EpiRAM device achieves the best accuracy of 84.5% even with a lower software baseline, which can be attributed to a slight asymmetry and a higher resiliency to cycle-to-cycle variations due to a relatively lower weight precision. Even with a good nonlinearity/symmetry, The TaO_x/HfO_x RRAM device shows a large degradation due to a large cycle-to-cycle variation. The HZO FeFET device shows a smallest accuracy degradation compared to the software baseline because of the best symmetry and the least cycle-to-cycle variation.

4.4 Summary

In this chapter, a comprehensive investigation on the impact of eNVM device nonidealities on training and inference is conducted with a PyTorch-based simulation framework, where the device nonidealities are modeled by generalized numerical models and incorporated into the DNN model. We perform the analysis with an virtual 8-bit device with assumed properties exhibited by typical eNVM devices. Our simulation results suggest that: (1)

the training accuracy is more sensitive to the asymmetry of conductance tuning than the nonlinearity, the high nonlinearity can be tolerated if the potentiation (P) and depression (D) have the same polarity while asymmetric P/D (which is typically the case in today's devices) significantly degrades the training accuracy; 2) the conductance range variation does not degrade the training accuracy significantly, instead, a small variation can even reduce the accuracy loss introduced by asymmetry; 3) D2D variations can also remedy the accuracy loss due to asymmetry while C2C variations lead to dramatic accuracy degradation; 4) The accuracy degradation will not be noticeable if the endurance cycles (defined as the number of programming cycles that cause the conductance tunable dynamic range decays by 50% are more than 7,000 cycles. 5) Different drifting modes affect the inference accuracy differently, the best case is where the conductance is drifting up/down randomly.

CHAPTER 5

2-TRANSISTOR-1-FEFET BASED WEIGHT CELL FOR TRAINING AND INFERENCE AT HYBRID PRECISION

5.1 Introduction

As we have observed in Chapter 4, the in-situ training with eNVMs suffers from unacceptable accuracy degradation due to various nonidealities including limited dynamic range, variation, and most importantly asymmetric conductance tuning behavior [43]. For example, we survey several analog eNVM based synapses for in-situ training in Figure 5.1. For filamentary RRAM [75], the excessive asymmetry/nonlinearity between positive and negative update leads to a poor accuracy 41% for MNIST dataset. While interfacial RRAM [69] exhibits improved nonlinearity with higher accuracy 73%, the programming pulse width is on the orders of millisecond due to the slow diffusion process of ions or vacancies. A recent discovery of partial polarization switching in ferroelectric-FET (FeFET) [45] provides highly symmetric weight update leading to an accuracy 90%, but non-identical pulses need to be applied for conductance tuning, which increases the peripheral circuitry complexity. Despite recent progress, these hardware implementations are not competitive with the software training accuracy 98% even for MNIST dataset.

To address the challenge, [46] proposed a capacitor-assisted cell design consisting of 3 transistors and 1 capacitor (3T1C), where a highly symmetric and linear weight update was achieved by modulating the gate voltage of storage transistor through two charging and discharging transistor while keeping the storage transistor working in the triode region. However, this is not efficient to be used for inference as the capacitor is volatile, meaning that the weight information needs to be backed up and restored during inference. Accordingly, [47] proposed a weight cell design that combines non-volatile PCM with volatile

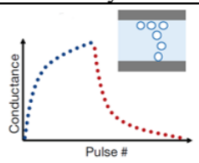
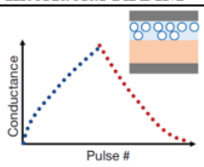
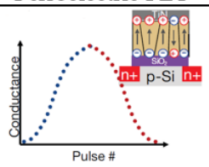
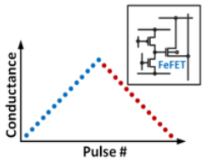
Analog Synapse Devices				
Type	Filamentary RRAM	Interfacial RRAM	Ferroelectric FET	This work
Weight update behavior				
Symmetry	Low	Medium	Medium	High
Programming	Identical Pulses	Identical Pulses	Non-identical Pulses	Identical Pulses
Speed	10-100ns	100µs-10ms	50ns-100ns	5ns
MNIST Accuracy	~41%	~73%	~90%	~97.3%

Figure 5.1: Comparison of analog synapses for on-chip in-situ learning. The proposed 2-Transistor-1FeFET design exhibits the desired characteristics including highly symmetric/linear weight update, fast and identical update pulses, allowing fast training of neural networks with high accuracy.

capacitor, where the incremental weight updates happen in the capacitor based 3T1C unit with symmetric and linear and tuning behavior and the weight information is transferred to non-volatile PCM cells for inference. With the proposed unit cell, they demonstrated a software-equivalent training and inference accuracy on various datasets. However, this is achieved at the expense of substantial area overhead, which limits the capacity for on-chip weight storage.

Motivated by the observation that in a DNN algorithm a relatively higher precision (larger than 6-bit) is necessary during training to accumulate the incremental weight change, while a lower precision (less than 2-bit) is sufficient during inference to achieve a reasonably good accuracy [97], we introduce a synaptic weight cell design in this work that combines two CMOS transistors and one FeFET (2T1F) for training and inference with hybrid precision. During training, the "volatile" modulated gate voltage of FeFET is used to represent LSBs for symmetric and linear update. This is achieved by modulating the FeFET gate voltage through charging and discharging pulses while keeping the FeFET working in the triode region. Occasionally, the information of LSBs will be transferred to MSBs to avoid errors due to the limited dynamic range of the gate voltage. After training process is complete, the information of LSBs is discarded, only MSBs are preserved by "non-

volatile" polarization states of FeFET for inference. We demonstrate a 6-bit/7-bit synapse design (2-bit MSBs + 4-bit/5-bit LSBs) for MNIST/CIFAR-10 dataset and benchmark with a LeNet-5-like/VGG-like CNN model. The SPICE simulation result with the experimental validated FeFET model and TSMC 65nm PDK is coupled with the CNN model set up in TensorFlow framework, showing that the learning accuracy could achieve 97%/ 87%, approaching the ideal software training.

5.2 2T-1FeFET Synaptic Weight Cell design

5.2.1 FeFET basics

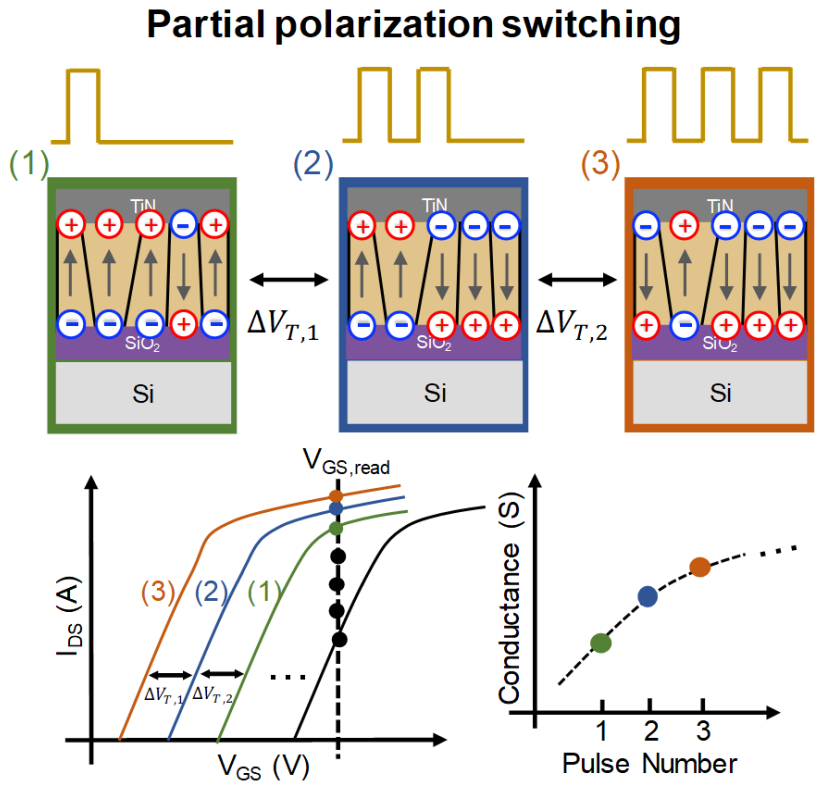


Figure 5.2: Gradual tuning of channel conductance of FeFET through partial polarization switching of the multi-domain in the ferroelectric layer. The gate capacitance and consequently the threshold voltage of the FeFET can be gradually tuned by the application of voltage pulses to the gate.

FeFET is a three-terminal device, which resembles a MOSTFET in structure, except a additional ferroelectric oxide layer is deposited in the gate-stack. Thanks to the much

improved CMOS compatibility and scalability, the HfO_2 thin film based FeFET has been under active research in recent years for potential applications in both storage and computing.The polarization state of the Si:HfO_2 thin film could be flipped by applying positive or negative electrical field across the thin film. Thus the gate capacitance and consequently the threshold voltage of the FeFET can be gradually tuned by applying corresponding voltage pulses to the gate. To program the cell, a positive gate voltage should be applied while grounding the substrate. Conversely, a negative gate voltage should be applied to erase the cell. Nevertheless,to avoid the on-chip generation of negative voltage, grounding the gate and apply positive voltage to the body would do the job as well. As demonstrated in Figure 5.2, with more programming pulses applied to the gate, more polarization domains are flipped, resulting a lower threshold voltage of the FeFET, thus a gradually tunable channel conductance is achieved. The work [45] experimentally demonstrate the tunable channel conductance with a $Hf_{0.5}Zr_{0.5}O_2$ (HZO) based FeFET. Several different pulse schemes are explored, including identical pulses, pulses with increasing pulse-width, and pulses with increasing amplitude. The results suggest that the non-identical pulse schemes can improve the linearity and symmetry of the tuning curve. In addition, as compared with RRAM, FeFET shows several better features such as larger ON/OFF ratio, shorter programming pulse-width, and less programming energy consumption.

5.2.2 2T1F Weight Cell Design

Figure 5.3 and Figure 5.4 show the schematic and fundamental principle of the proposed 2T1F synaptic weight cell. The weight cell consists of one FeFET, one pull-up PMOS and one pull-down NMOS. The FeFET gate capacitor serves as an analog storage memory for LSBs, which is charged/discharged by the corresponding pull-up PMOS and pull-down NMOS. Thus, the LSBs of the weight information can be encoded to the channel conductance of the FeFET by modulating the gate voltage (V_G) while keeping the FeFET working in the triode region as shown in Figure 5.4(a). During weight update, positive/negative

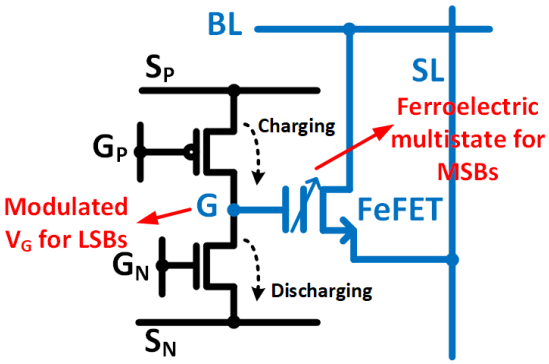


Figure 5.3: Schematic of the proposed 2T1F weight cell design.

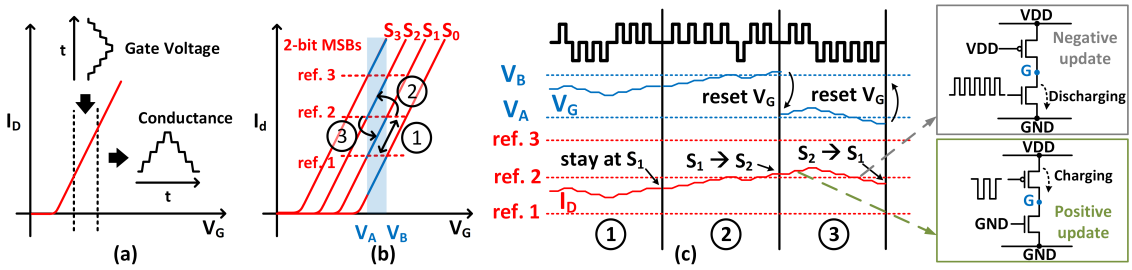


Figure 5.4: (a) The LSBs of weight are linearly encoded to the conductance value of the FeFET by modulating the gate voltage while keeping the FeFET in the triode region. (b) The MSBs are encoded to different FeFET polarization states without overlapping of LSBs within each MSB state. (c) Illustration of updating LSBs within a FeFET polarization state and updating MSBs depending on the corresponding read-out current level.

updates trigger the charging/discharging pulses to be applied to the gate of FeFET while keeping these two transistors working in saturation region to make the charging/discharging current less independent of V_G. By biasing the pulse amplitudes to generate the balanced charging and discharging current, the positive/negative updates of LSBs are expected to be linear and symmetric. The MSBs of the weight information are encoded to different FeFET polarization states without overlapping LSBs within each MSB state. For example, assuming 2-bit MSBs (i.e., 4 polarization states) as shown in Figure 5.4(b), the V_G dynamic range (V_A, V_B) which is constrained by the linear region overlap of multiple polarization states, and the voltage step size, which is controlled by the pulse amplitude and width, determine the number of LSB update steps (i.e., the bitwidth of LSBs). Figure 5.4(c) illustrates 3 different weight update scenarios of LSBs and MSBs. First, if V_G stays inside

the dynamic range, i.e., the weight change is relatively small, nothing will be transferred to FeFET states. If V_G increases beyond V_B, the consequential read-out current I_D will be larger than the reference current (ref. 2 in Figure 5.4(c)), which will trigger a FeFET programming process towards S2 state to transfer the weight information to MSBs, then the LSBs can be continuously updated within S2 state and V_G needs to be reset to the certain level that maintains the same I_D to prevent the error of LSBs. Similarly, if I_D decreases below ref. 2, the FeFET need to be programmed towards S1 state, together with a reset operation to the gate voltage V_G. Thus, a continuous and symmetric weight update process can be achieved.

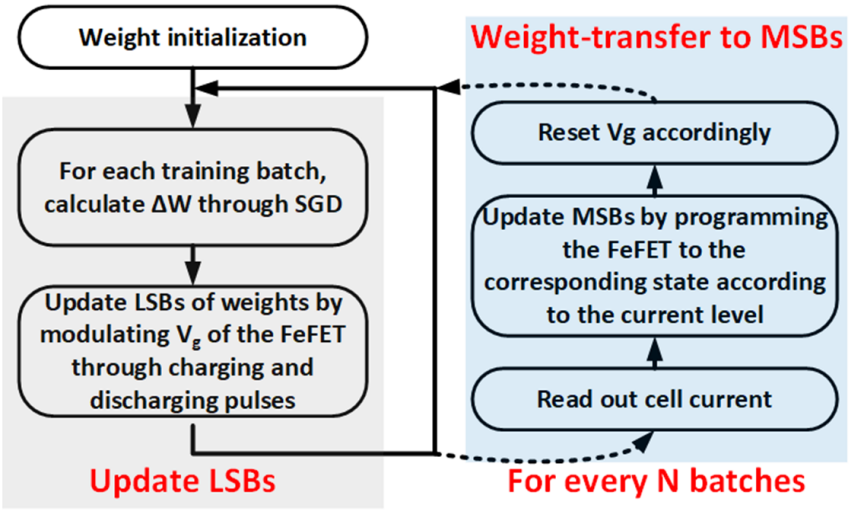


Figure 5.5: The training flow chart. For each training batch, update LSBs by applying charging/discharging pulses to modulate V_G based on the value of ΔW. For every N batches, program the FeFET to the corresponding polarization state according to the read-out current level, namely weight-transfer.

With the proposed synaptic weight cell design, we customize the DNN training flow as shown in Figure 5.5. During each training batch, LSBs are updated by applying charging/discharging pulses to modulate V_G according to the value of ΔW calculated through stochastic gradient descent (SGD) based backpropagation algorithm. However, the information of LSBs needs to be occasionally transferred to MSBs to prevent the information loss due to the limited V_G dynamic range and capacitor leakage,. Therefore, after a cer-

tain number of bathces, we need to perform the weight-transfer, i.e., programming the FeFET to the corresponding state according to the read-out current level. After the weight-transfer,V_G prefers to be reset to the certain level that maintains the same cell current to recover the residual information of LSBs. Nevertheless, this step requires a high-precision ADC (equals the total bitwidth of weights) which induces much power and area overhead. Therefore, we only reset V_G to $(V_A+V_B)/2$ to avoid high-precision ADCs at the expense of inducing potential residual errors. The impact of these residual errors on learning accuracy is investigated in the evaluation section.

5.2.3 Implementation of 2-bit MSBs + 4-bit LSBs Synapse

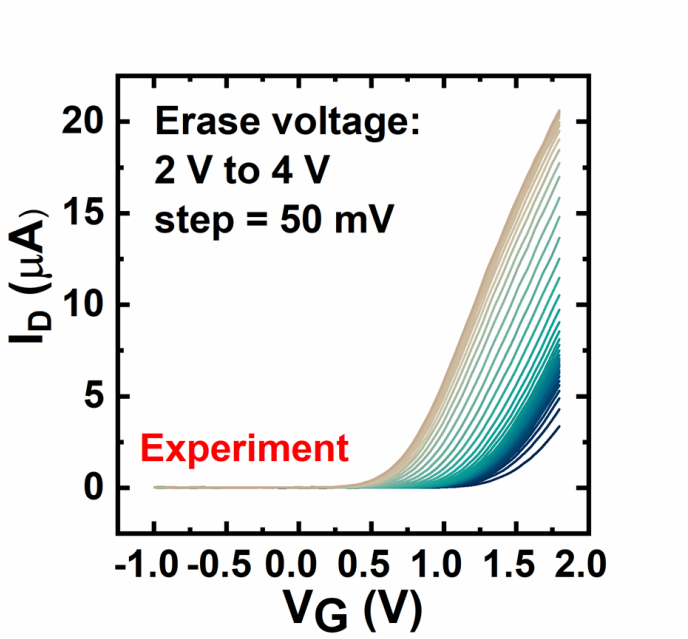


Figure 5.6: Measured I_D-V_G characteristics of a fabricated HZO FeFET [111] for programming voltage from 2V to 4V, showing tunable threshold voltages.

In this section, we demonstrate the implementation of 6-bit synapse (2-bit MSBs + 4-bit LSBs) as an example. Figure 5.6 shows the measured I_D-V_G characteristics of a fabricated HZO FeFET with tunable V_{th}. It can be seen that different threshold voltages are achieved with different programming voltages. It is worth mentioning that our hybrid-precision design can much relax the requirement on the number of analog states of FeFET,

thus improving the resiliency to device variations. We adopt the FeFET SPICE model from the prior work [111], where the model consists of a conventional MOSFET model based on BSIM-4, and a ferroelectric switching model based on Preisach dynamic model. The SPICE model accurately captures the experimental P-V loop as shown in Figure 5.7.

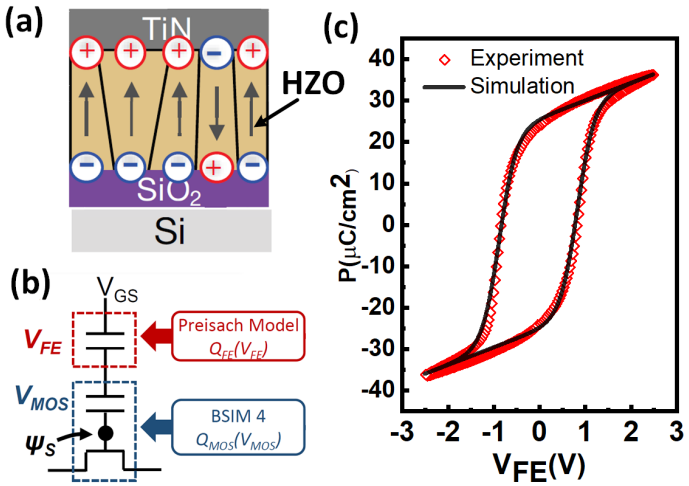


Figure 5.7: (a) Schematic of partially switching of HZO ferroelectric domains. (b) The FeFET model [111] consists of the conventional MOSFET modeled by BSIM-4 and the ferroelectric layer modeled by the dynamic Preisach model. (c) The model accurately captures the experimental P-V loop.

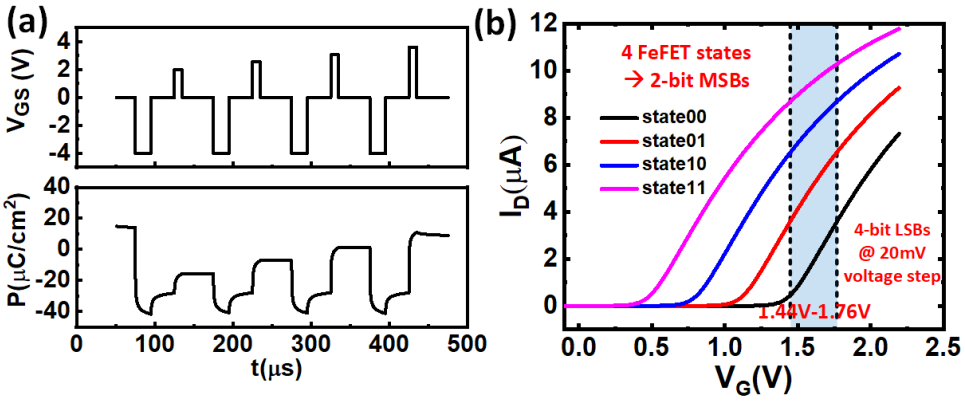


Figure 5.8: (a) The pulse scheme and the corresponding remnant polarization charge that generates 4 FeFET states. (b) Simulated I_D vs. V_G curve of different FeFET states. 4 polarization states serve as 2-bit MSBs. The dynamic range of V_G is set to be $[1.44V, 1.76V]$, with a pulse width of 5 ns that leads to ΔV_G of 20 mV per update pulse, thus 4-bit LSBs can be achieved.

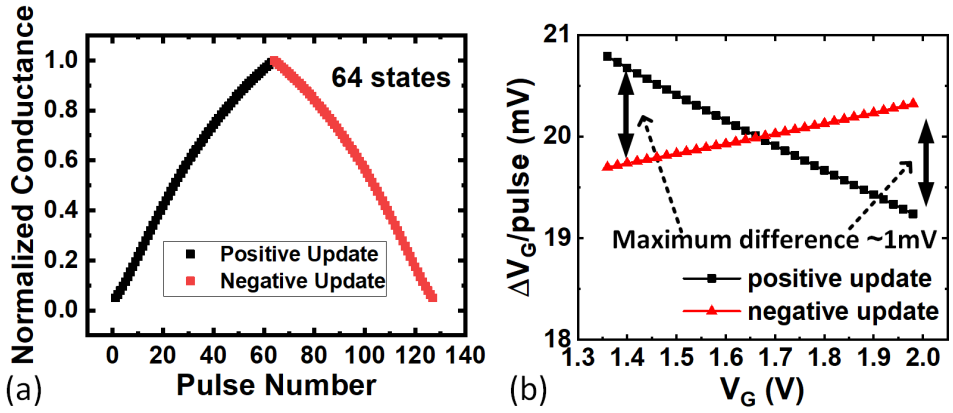


Figure 5.9: (a) The equivalent conductance update curve of the 6-bit synapse realized by 2T1F weight cell, showing much improved symmetry and linearity between positive update and negative update. (b) ΔV_G per pulse during positive update and negative update as a function of V_G. The maximum difference of ΔV_G between two directions is only 1 mV (5% of one LSB step), suggesting symmetry in weight update.

With the SPICE model, we tune the programming condition to generate 4 dedicated polarization states to represent 2-bit MSBs. Figure 5.8(a) shows the pulse scheme and the simulated corresponding remnant polarization charge that result in 4 states shown inFigure 5.8(b), which serve as 2-bit MSBs. To ensure that 4 dedicated states can be accurately reached without being affected by the history effect, a strong reset pulse is applied to fully erase the state before applying the set pulse to program it to the corresponding state. Given the simulated I_D-V_G characteristic, the dynamic range of V_G is set to be $[1.44V, 1.76V]$, with a pulse width of 5 ns that leads to a step voltage size ΔV_G of 20 mV for each update pulse, thus 4-bit LSBs can be achieved within each MSB state.

The equivalent weight update curve of the 6-bit synapse is shown in Figure 5.9(a). 64 continuous conductance states are achieved with bidirectional tunability. However, as the charging/discharging current cannot remain perfectly unchanged as V_G changes, ΔV_G per update pulse is not perfectly the same at different V_G, resulting a slight nonlinearity as shown in Figure 5.9(b). Nevertheless, the weight update is still symmetric as the maximum difference of ΔV_G between positive update and negative update is only 5% of one LSB step, which will not cause LSB error during weight update.

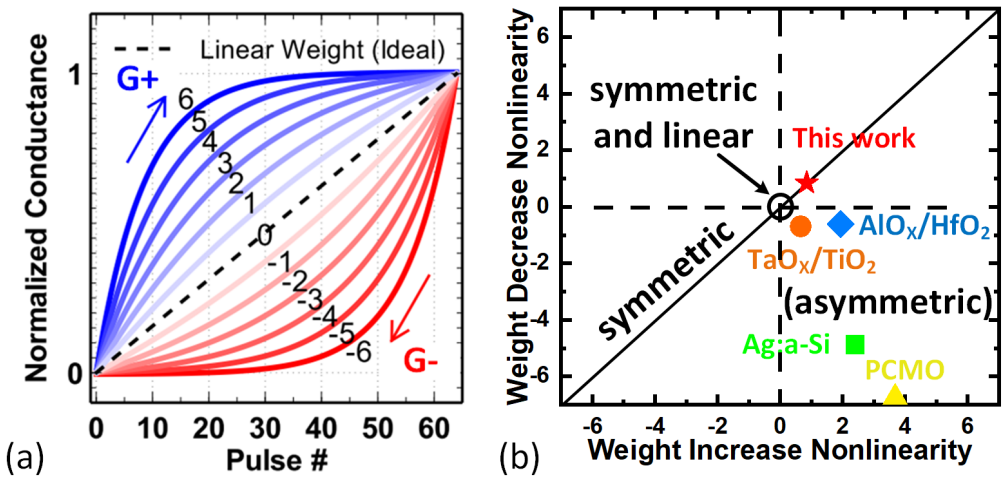


Figure 5.10: (a) Analog NVM device behavioral model [98] of the nonlinear/asymmetric weight update. The nonlinearity degree is labeled from $+6$ to -6. (b) Comparison of the asymmetry/linearity between this work and other pure eNVM devices [98].

To evaluate the degree of nonlinearity and asymmetry of various synapse candidates, we exploited the mathematical model developed in [98] as shown in Figure 5.10(a). For example, linearity of 0 means the conductance tuning is ideally linear and symmetric while the larger the factor, the worse the linearity. The sign of the factor represents the direction of the conductance tuning, i.e., weight increase or weight decrease. Figure 5.10(b) shows the nonlinearity factors of various standalone eNVM devices, which are all bearing nonlinearity and asymmetry to a certain degree [98]. This work falls on the diagonal of the plot with the coordinate of (1,1), meaning that the conductance tuning behavior is symmetric while bearing a slight nonlinearity.

5.3 Simulation Results and Discussion

We benchmark the performance of the proposed hybrid 6-bit 2T1F synapse by incorporating the aforementioned synaptic characteristics into TensorFlow simulation with a CNN, which is a variation of LeNet-5 as shown in Figure 5.11(a), on MNIST dataset. The network consists of two convolutional layers with 5×5 kernels and two fully connected layers. The input digit image dimension is 28×28 pixels. We train the model for 300 epochs with

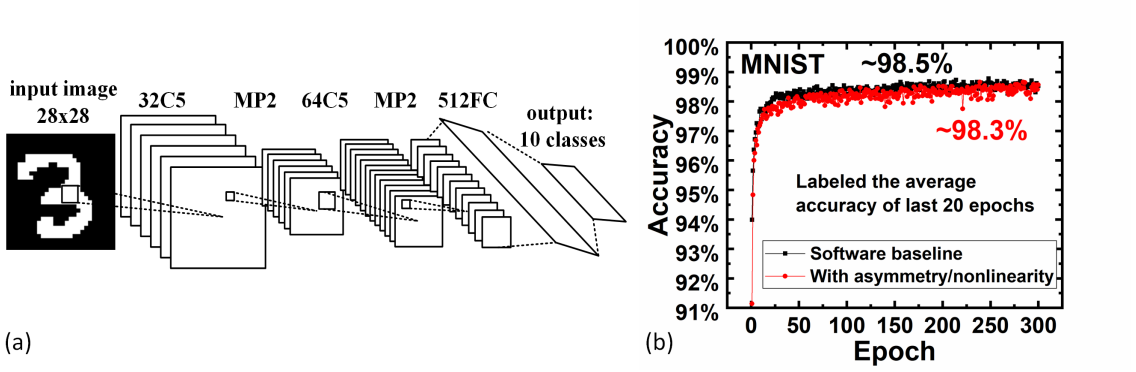


Figure 5.11: (a) Evaluated with a CNN model on MNIST dataset. The adopted CNN is a variation of LeNet-5 with 32C5-MP2-64C5-MP2-512FC-10 configuration. (b) The MNIST learning accuracy can achieve 98.3% with the slight nonlinearity of the proposed 2T1F design, showing only 0.2% degradation compared to the ideal software training with 6-bit weights.

batch size of 100 images. The baseline training accuracy is ∼98.5% from ideal software training with 6-bit weights. With the extracted 6-bit weight cell, the training accuracy only drops ∼0.2% to 98.3% as shown in Figure 5.11(b).

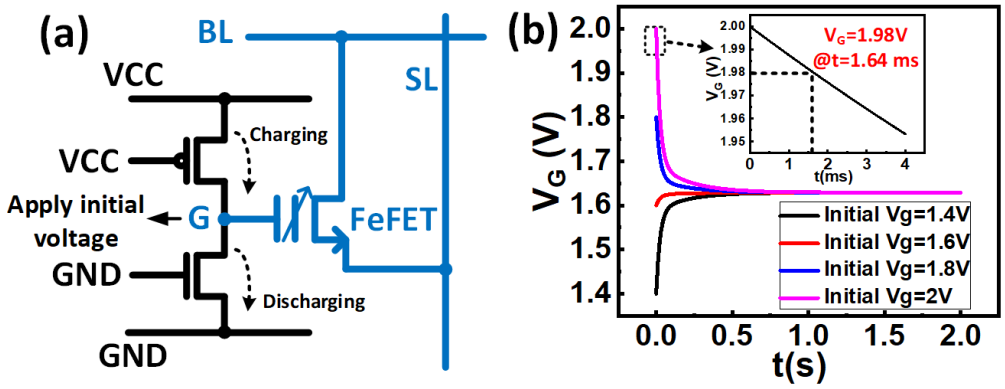


Figure 5.12: (a) Circuit setup for leakage simulation. (b) Simulation results of V_G leakage with different starting values of V_G. The inset figure shows that it takes 1.64 ms for V_G to leak by one LSB step (20 mV) in the worst case (starting V_G = 2 V). Assuming the training time is $\sim 7\mu s$ per batch, the maximum transfer interval becomes ∼230 batches, limited by the leakage.

Then we investigate the impact of residual errors caused by occasional weight-transfer on the training accuracy. First, we analyze the leakage of the FeFET gate voltage due to the off-current of PMOS/NMOS. Figure 5.12 shows the simulation results of V_G leakage

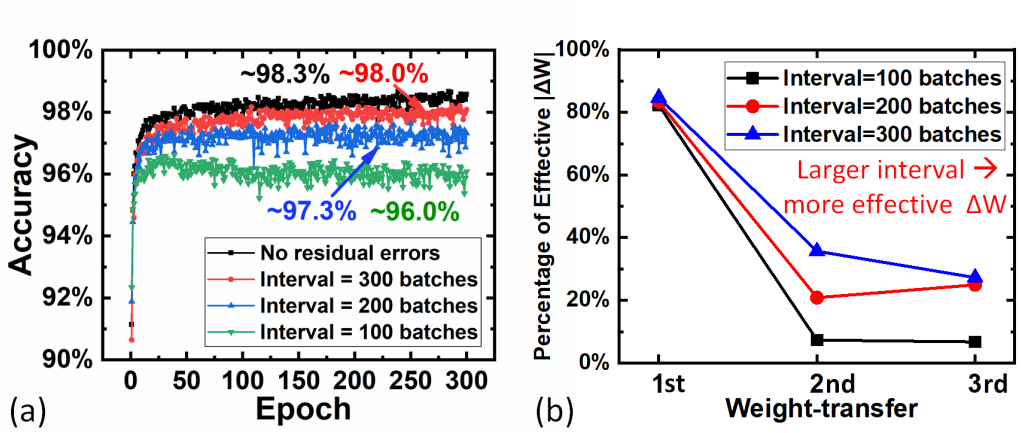


Figure 5.13: (a) The MNIST learning accuracy with weight-transfer interval of 100, 200, and 300 batches, achieving 96.0%, 97.3%, and 98.0% respectively. (b) The percentage of effective $\mid \Delta W \mid$ (> 8 LSB steps) during first, second, and third weight-transfer with different number of interval batches. A larger interval leads to a larger percentage of effective $\mid \Delta W \mid$ to be accumulated, which benefits the training.

with different starting values of V_G, the inset figure shows that it takes 1.64 ms for V_G to drift by one LSB step (20 mV) in the worst case (starting V_G = 2 V). Then we estimate the training time per batch assuming that the array size is 128×128, very 8 columns share one read circuit (i.e., ADC), and read-out latency is 1 ns. Meanwhile, we configure the training algorithm to make sure that the weight update value is at most 1 LSB step per batch. Overall, the training time per batch (forward + backward + update, batch size is 100) is estimated to be $\sim 7\mu s$, thus the maximum transfer interval becomes ~230 batches to avoid any residual errors.

We analyze the impact of the value of transfer-interval on the training accuracy. Figure 5.13(a) shows the training accuracy results with transfer interval of 100, 200, and 300 batches respectively. When the transfer interval is 100 batches, the accuracy can only achieve ~96%, when the transfer interval is increased to 300 batches, the accuracy can reach ~98.0%, showing a negligible degradation compared to 98.3%. The reason is that if the absolute accumulated ΔW within one transfer interval is less than half of one MSB step (8 LSB steps), which fails to trigger the MSBs state change, the weight will be reset back after weight transfer as the V_G will be reset to $(V_A+V_B)/2$ as aforementioned. Fig-

ure 5.13(b) shows the percentage of effective $\mid \Delta W \mid$ (>8 LSB steps) during first, second, and third weight-transfer operations as an example. A larger interval leads to a larger percentage of effective $\mid \Delta W \mid$. Given the fact that weights tend to be stabilized through training process, i.e., the value of $\mid \Delta W \mid$ is expected to decay over time, a dynamic transfer interval (increasing through training) is preferred to trigger more effective updates to reduce the accuracy degradation.

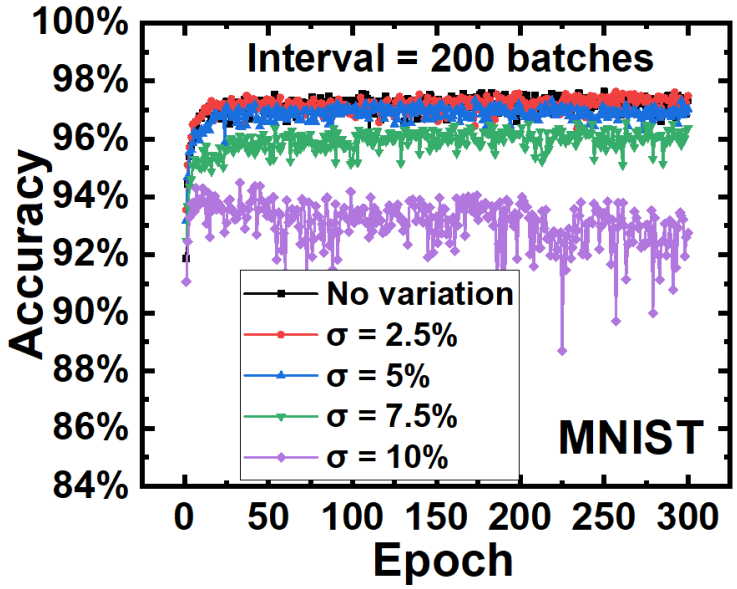


Figure 5.14: The impact of FeFET polarization state variation on the MNIST learning accuracy. A small variation does not hurt the accuracy as it may help on compensating the residual errors caused by non-ideal weight-transfer. The degradation becomes unacceptable when variation exceeds 5%.

Figure 5.14 shows the impact of FeFET polarization state variation on the learning accuracy. A small variation (<2.5%) does not hurt the accuracy as it may help on compensating the residual errors caused by non-ideal weight-transfer. The degradation becomes unacceptable when variation exceeds 5%.

We also perform the accuracy evaluation on CIFAR-10 dataset with a VGG-like CNN, which consist of 6 convolutional layers and 2 fully connected layers. Since CIFAR-10 dataset is more complex that MNIST dataset, we extend the weight cell precision to 7-bit

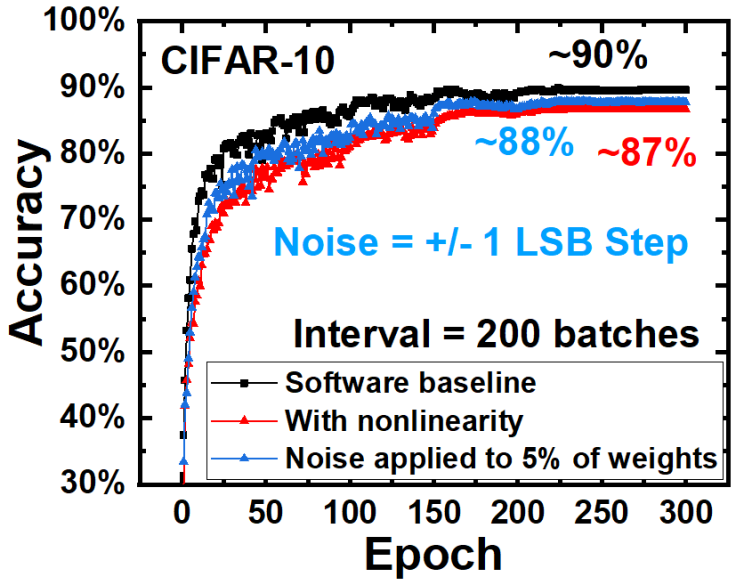


Figure 5.15: The learning accuracy on CIFAR-10 dataset could achieve 87% without noise and 88% with noise using the proposed 7-bit synapse with a VGG-like CNN.

by reducing the LSB tuning step to 10 mV to enable 5-bit LSBs. Meanwhile, we consider the potential variation on the LSB update introduced by the noise. Figure 5.15 shows that the accuracy can achieve ∼87% without noise, and ∼88% with noise of one LSB step added to 5% of the total weights due to the random fluctuation of a 10mV step in practice. We speculate that the noise may also help on the compensation of the residual errors, which could be a future work to explore.

Table 5.1 compares this work with recent works with "volatile" capacitor-assisted designs. The work [46] using 3T1C is totally volatile thus not suitable for offline inference. While the work [47] enables both training and inference by combing 3T1C and 2 PCM cells, it has relatively higher programming energy compared to our 2T1F design.

5.4 Summary

In this work, to overcome the challenges posed by the typical nonlinear and asymmetric conductance tuning characteristic of eNVMs, we propose a synaptic weight cell design

Table 5.1: Comparison with Prior Capacitor-Assisted Works

Work	[46]	[47]	This work
Weight cell	3T1C	2PCM+3T1C	2T1F
Programming energy	Low	High	Medium
Training	Yes	Yes	Yes
Inference	No	Yes	Yes

that combines two CMOS transistors and one FeFET (2T1F) for training and inference with hybrid precision. During training, the "volatile" modulated gate voltage of FeFET is used to represent LSBs for symmetric and linear update. After training process is done, the information of LSBs is discarded, only MSBs are preserved by "non-volatile" polarization states of FeFET for inference. We demonstrate a 6-bit/7-bit synapse design (2-bit MSBs + 4-bit/5-bit LSBs) for MNIST/CIFAR-10 dataset and benchmark with a LeNet-5-like/VGG-like CNN. The SPICE simulation result with the experimentally validated FeFET model and TSMC 65nm CMOS PDK is coupled with the model set up in TensorFlow framework, showing that the learning accuracy could achieve $\sim$97.3%/$\sim$87%, approaching the ideal software training.

CHAPTER 6
CONCLUSION

6.1 Key Contributions

In this research, we first focus on the eNVM-based inference acceleration and accomplish pioneering works on RRAM-based CIM accelerator design. Two generations of RRAM testchips which monolithically integrate the RRAM memory array and CMOS peripheral circuits are designed and fabricated using Winbond 90 nm and TSMC 40 nm commercial embedded RRAM process respectively. The 1^{st} generation of testchip named XNOR-RRAM is dedicated for binary neural networks (BNNs) and the 2^{nd} generation named Flex-RRAM features 1bit-to-8bit run-time configurable precision and leverages the input sparsity of the DNN model to improve the throughput and energy-efficiency.

However, the non-ideal characteristics of eNVM devices, especially when utilized as multi-level analog synaptic weights, may incur a notable accuracy degradation for both training and inference. This research develops a PyTorch based framework that incorporates the device characteristics into the DNN model to evaluate the impact of the eNVM nonidealities on training/inference accuracy. The results suggest that it is challenging to directly use eNVMs for in-situ training and resistance drift remains as a critical challenge to maintain a high inference accuracy.

Furthermore, to overcome the challenges posed by the asymmetric conductance tuning behavior of typical eNVMs, which is found to be the most critical nonideality that prevents the model from achieving software-equivalent training accuracy, this research proposes a novel 2-transistor-1-FeFET (ferroelectric field effect transistor) based synaptic weight cell that exploits hybrid precision for in-situ training and inference, which achieves near-software classification accuracy on MNIST and CIFAR-10 dataset.

6.2 Future Works

While eNVM-based CIM accelerators have demonstrated unprecedented efficiency over conventional digital counterparts, they are still premature to be applied to practical products due to some missing parts from device characteristics up to architectural considerations. At device level, the device with ideal characteristic in terms of multi-level programmability, reliability, and uniformity is still missing. The advances in device engineering are highly desired. At circuit level, as the most critical unit in typical CIM system, ADCs remain as the bottleneck in terms of area and power consumption. Given that there may be little space for further constraining the area and power of ADCs while maintaining a acceptable performance, 3D IC stacking could be an attractive solution to mitigate the ADC bottleneck as it can spare more space to accommodate ADCs. At architecture level, most of the chip-level demonstrations so far are focused on small-scale core design. A complete system with a sufficient number of cores and the related interconnects is missing and highly desired. These open questions at different abstraction levels present as promising future directions.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, *et al.*, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 8604–8608.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[7] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[8] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[9] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE micro*, vol. 30, no. 2, pp. 56–69, 2010.

[10] J. T. Pawlowski, "Hybrid memory cube (HMC)," in *2011 IEEE Hot chips 23 symposium (HCS)*, IEEE, 2011, pp. 1–24.

[11] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, *et al.*, "25.2 a 1.2 V 8Gb 8-channel 128GB/s high-bandwidth memory (hbm) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV," in *2014 IEEE International Solid-State*

Circuits Conference Digest of Technical Papers (ISSCC), IEEE, 2014, pp. 432–433.

[12] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12.

[13] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE, 2014, pp. 609–622.

[14] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, 2015, pp. 92–104.

[15] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 367–379, 2016.

[16] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 75–84.

[17] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[18] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[19] W.-S. Khwa, J.-J. Chen, J.-F. Li, X. Si, E.-Y. Yang, X. Sun, R. Liu, P.-Y. Chen, Q. Li, S. Yu, *et al.*, "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 496–498.

[20] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 488–490.

[21] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, *et al.*, "24.5 a twin-8t SRAM computation-in-memory macro for multiple-bit cnn-based machine learning," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 396–398.

[22] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[23] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

[24] J.-W. Su, X. Si, Y.-C. Chou, T.-W. Chang, W.-H. Huang, Y.-N. Tu, R. Liu, P.-J. Lu, T.-W. Liu, J.-H. Wang, *et al.*, "15.2 a 28nm 64kb inference-training two-way transpose multibit 6t SRAM compute-in-memory macro for AI edge chips," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 240–242.

[25] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "15.3 a 351TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 242–244.

[26] P. Jain, U. Arslan, M. Sekhar, B. C. Lin, L. Wei, T. Sahu, J. Alzate-vinasco, A. Vangapaty, M. Meterelliyoz, N. Strutt, *et al.*, "13.2 a 3.6 Mb 10.1 Mb/mm 2 embedded non-volatile ReRAM macro in 22nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 v with sensing time of 5ns at 0.7 V," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 212–214.

[27] C.-C. Chou, Z.-J. Lin, P.-L. Tseng, C.-F. Li, C.-Y. Chang, W.-C. Chen, Y.-D. Chih, and T.-Y. J. Chang, "An n40 256k$\times$ 44 embedded RRAM macro with sl-precharge SA and low-voltage current limiter to improve read and write performance," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 478–480.

[28] L. Wei, J. G. Alzate, U. Arslan, J. Brockman, N. Das, K. Fischer, T. Ghani, O. Golonzka, P. Hentges, R. Jahan, *et al.*, "13.3 a 7mb STT-SRAM in 22FFL FinFET technology with 4 ns read sensing time at 0.9 v using write-verify-write scheme and offset-cancellation sensing technique," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 214–216.

[29] Y. Song, J. Lee, S. Han, H. Shin, K. Lee, K. Suh, D. Jeong, G. Koh, S. Oh, J. Park, *et al.*, "Demonstration of highly manufacturable STT-SRAM embedded in 28nm

logic," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 18–2.

[30] J. Wu, Y. Chen, W. Khwa, S. Yu, T. Wang, J. Tseng, Y. Chih, and C. H. Diaz, "A 40nm low-power logic compatible phase change memory technology," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 27–6.

[31] S. Dünkel, M. Trentzsch, R. Richter, P. Moll, C. Fuchs, O. Gehring, M. Majer, S. Wittek, B. Müller, T. Melde, *et al.*, "A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 19–7.

[32] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, *et al.*, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 494–496.

[33] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang, *et al.*, "24.1 a 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 388–390.

[34] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, *et al.*, "15.4 a 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 244–246.

[35] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, *et al.*, "33.2 a fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel MAC computing," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2020, pp. 500–502.

[36] S. Yin, Y. Kim, X. Han, H. Barnaby, S. Yu, Y. Luo, W. He, X. Sun, J.-J. Kim, and J.-s. Seo, "Monolithically integrated RRAM-and CMOS-based in-memory computing optimizations for efficient deep learning," *IEEE Micro*, vol. 39, no. 6, pp. 54–63, 2019.

[37] C. Ho, S.-C. Chang, C.-Y. Huang, Y.-C. Chuang, S.-F. Lim, M.-H. Hsieh, S.-C. Chang, and H.-H. Liao, "Integrated HfO_2-RRAM to achieve highly reliable, greener, faster, cost-effective, and scaled devices," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 2–6.

[38] J. Park, M. Kwak, K. Moon, J. Woo, D. Lee, and H. Hwang, "TiO_x-based RRAM synapse with 64-levels of conductance and symmetric conductance change by adopting a hybrid pulse scheme for neuromorphic computing," *IEEE Electron Device Letters*, vol. 37, no. 12, pp. 1559–1562, 2016.

[39] A. Prakash, J. Park, J. Song, J. Woo, E.-J. Cha, and H. Hwang, "Demonstration of low power 3-bit multilevel cell characteristics in a TaO_x-based RRAM by stack engineering," *IEEE Electron Device Letters*, vol. 36, no. 1, pp. 32–34, 2014.

[40] T. Nirschl, J. Philipp, T. Happ, G. W. Burr, B. Rajendran, M.-H. Lee, A. Schrott, M. Yang, M. Breitwisch, C.-F. Chen, *et al.*, "Write strategies for 2 and 4-bit multilevel phase-change memory," in *2007 IEEE International Electron Devices Meeting*, IEEE, 2007, pp. 461–464.

[41] S. Kim, B. Lee, M. Asheghi, F. Hurkx, J. P. Reifenberg, K. E. Goodson, and H.-S. P. Wong, "Resistance and threshold switching voltage drift behavior in phase-change memory and their temperature dependence at microsecond time scales studied using a micro-thermal stage," *IEEE Transactions on Electron Devices*, vol. 58, no. 3, pp. 584–592, 2011.

[42] D. Ielmini, S. Lavizzari, D. Sharma, and A. L. Lacaita, "Physical interpretation, modeling and impact on phase change memory (pcm) reliability of resistance drift due to chalcogenide structural relaxation," in *2007 IEEE International Electron Devices Meeting*, IEEE, 2007, pp. 939–942.

[43] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 570–579, 2019.

[44] W. Wu, H. Wu, B. Gao, P. Yao, X. Zhang, X. Peng, S. Yu, and H. Qian, "A methodology to improve linearity of analog RRAM for neuromorphic computing," in *2018 IEEE Symposium on VLSI Technology*, IEEE, 2018, pp. 103–104.

[45] M. Jerry, P.-Y. Chen, J. Zhang, P. Sharma, K. Ni, S. Yu, and S. Datta, "Ferroelectric FET analog synapse for acceleration of deep neural network training," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 6–2.

[46] Y. Li, S. Kim, X. Sun, P. Solomon, T. Gokmen, H. Tsai, S. Koswatta, Z. Ren, R. Mo, C. Yeh, *et al.*, "Capacitor-based cross-point array for analog neural network with record symmetry and linearity," in *2018 IEEE Symposium on VLSI Technology*, IEEE, 2018, pp. 25–26.

[47] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. Farinha, *et al.*, "Equivalent-accuracy acceler-

ated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, 2018.

[48] S. Huang, X. Sun, X. Peng, H. Jiang, and S. Yu, "Overcoming challenges for achieving high in-situ training accuracy with emerging memories," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2020, pp. 1025–1030.

[49] X. Sun, P. Wang, K. Ni, S. Datta, and S. Yu, "Exploiting hybrid precision for training and inference: A 2T-1FeFET based analog synaptic weight cell," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 3–1.

[50] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, no. 34, p. 14, 1998.

[51] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[52] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[53] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, classifiaction," 1992.

[54] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[55] T. Mikolov, S. Kombrink, L. Burget, J. Černock, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2011, pp. 5528–5531.

[56] G. Castellano, A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE transactions on Neural networks*, vol. 8, no. 3, pp. 519–531, 1997.

[57] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[60] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[61] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.

[62] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, *et al.*, "A 20nm 1.8 v 8Gb PRAM with 40MB/s program bandwidth," in *2012 IEEE International Solid-State Circuits Conference*, IEEE, 2012, pp. 46–48.

[63] P.-Y. Chen, "Design of resistive synaptic devices and array architectures for neuromorphic computing," PhD thesis, Arizona State University, 2018.

[64] C. Ho, E. Lai, M. Lee, C. Pan, Y. Yao, K. Hsieh, R. Liu, and C.-Y. Lu, "A highly reliable self-aligned graded oxide WO_x resistance memory: Conduction mechanisms and reliability," in *2007 IEEE Symposium on VLSI Technology*, IEEE, 2007, pp. 228–229.

[65] H. Lee, P. Chen, T. Wu, Y. Chen, C. Wang, P. Tzeng, C. Lin, F. Chen, C. Lien, and M.-J. Tsai, "Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO_2 based RRAM," in *2008 IEEE International Electron Devices Meeting*, IEEE, 2008, pp. 1–4.

[66] C. Rohde, B. J. Choi, D. S. Jeong, S. Choi, J.-S. Zhao, and C. S. Hwang, "Identification of a determining parameter for resistive switching of TiO_2 thin films," *Applied Physics Letters*, vol. 86, no. 26, p. 262 907, 2005.

[67] Y. Wu, B. Lee, and H.-S. P. Wong, "Ultra-low power Al_2O_3-based RRAM with 1μa reset current," in *Proceedings of 2010 International Symposium on VLSI Technology, System and Application*, IEEE, 2010, pp. 136–137.

[68] Z. Wei, Y. Kanzawa, K. Arita, Y. Katoh, K. Kawai, S. Muraoka, S. Mitani, S. Fujii, K. Katayama, M. Iijima, *et al.*, "Highly reliable TaO_x ReRAM and direct evidence of redox reaction mechanism," in *2008 IEEE International Electron Devices Meeting*, IEEE, 2008, pp. 1–4.

[69] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[70] I.-T. Wang, Y.-C. Lin, Y.-F. Wang, C.-W. Hsu, and T.-H. Hou, "3d synaptic architecture with ultralow sub-10 fj energy per spike for neuromorphic computation," in *2014 IEEE International Electron Devices Meeting*, IEEE, 2014, pp. 28–5.

[71] L. Gao, I.-T. Wang, P.-Y. Chen, S. Vrudhula, J.-s. Seo, Y. Cao, T.-H. Hou, and S. Yu, "Fully parallel write/read in resistive synaptic array for accelerating on-chip learning," *Nanotechnology*, vol. 26, no. 45, p. 455 204, 2015.

[72] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. Lee, B. Lee, and H.-J. Hwang, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *2013 IEEE International Electron Devices Meeting*, IEEE, 2013, pp. 25–6.

[73] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation," *Advanced Materials*, vol. 25, no. 12, pp. 1774–1779, 2013.

[74] W. Wu, H. Wu, B. Gao, N. Deng, S. Yu, and H. Qian, "Improving analog switching in HfO_x-based resistive memory with a thermal enhanced layer," *IEEE Electron Device Letters*, vol. 38, no. 8, pp. 1019–1022, 2017.

[75] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, "Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Letters*, vol. 37, no. 8, pp. 994–997, 2016.

[76] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.

[77] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *2011 International Electron Devices Meeting*, IEEE, 2011, pp. 4–4.

[78] D. Kuzum, R. G. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano letters*, vol. 12, no. 5, pp. 2179–2186, 2012.

[79] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[80] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong, *et al.*, "Face classification using electronic synapses," *Nature communications*, vol. 8, no. 1, pp. 1–8, 2017.

[81] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 Mb rram macro chip for classification and online training," in *2016 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2016, pp. 16–2.

[82] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, *et al.*, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature communications*, vol. 9, no. 1, pp. 1–8, 2018.

[83] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, and Y. Gohou, "A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in *2018 IEEE Symposium on VLSI Technology*, IEEE, 2018, pp. 175–176.

[84] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290–299, 2019.

[85] "Yale face database," *URL http://vision.ucsd.edu/content/yale-face-database*,

[86] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[87] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*, Springer, 2016, pp. 525–542.

[88] C.-P. Lo, W.-Z. Lin, W.-Y. Lin, H.-T. Lin, T.-H. Yang, Y.-N. Chiang, Y.-C. King, C.-J. Lin, Y.-D. Chih, T.-Y. J. Chang, *et al.*, "Embedded 2Mb ReRAM macro with 2.6 ns read access time using dynamic-trip-point-mismatch sampling current-mode sense amplifier for ioe applications," in *2017 Symposium on VLSI Circuits*, IEEE, 2017, pp. C164–C165.

[89] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, 1960.

[90] S. Yin, X. Sun, S. Yu, and J.-s. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated rram and 90nm cmos," *arXiv preprint arXiv:1909.07514*, 2019.

[91] X. Sun, S. Yin, X. Peng, R. Liu, J.-s. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 1423–1428.

[92] C.-Y. Chen, M. Q. Le, and K. Y. Kim, "A low power 6-bit flash ADC with reference voltage and common-mode calibration," *IEEE Journal of solid-state circuits*, vol. 44, no. 4, pp. 1041–1046, 2009.

[93] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[94] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[95] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 2861–2865.

[96] D. Kadetotad, Z. Xu, A. Mohanty, P.-Y. Chen, B. Lin, J. Ye, S. Vrudhula, S. Yu, Y. Cao, and J.-s. Seo, "Parallel architecture with resistive crosspoint array for dictionary learning acceleration," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 194–204, 2015.

[97] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[98] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018.

[99] L. Gao, P.-Y. Chen, and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," *IEEE Electron Device Letters*, vol. 36, no. 11, pp. 1157–1159, 2015.

[100] J. Woo and S. Yu, "Resistive memory-based analog synapse: The pursuit for linear and symmetric weight update," *IEEE Nanotechnology Magazine*, vol. 12, no. 3, pp. 36–44, 2018.

[101] I.-T. Wang, C.-C. Chang, L.-W. Chiu, T. Chou, and T.-H. Hou, "3D Ta/TaO_x/TiO_2/Ti synaptic array and linearity tuning of weight update for hardware neural network applications," *Nanotechnology*, vol. 27, no. 36, p. 365 204, 2016.

[102] P.-Y. Chen and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance," in *2018 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2018, pp. 5C–4.

[103] M. Zhao, H. Wu, B. Gao, X. Sun, Y. Liu, P. Yao, Y. Xi, X. Li, Q. Zhang, K. Wang, *et al.*, "Characterizing endurance degradation of incremental switching in analog rram for neuromorphic systems," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 20–2.

[104] M. Zhao, H. Wu, B. Gao, Q. Zhang, W. Wu, S. Wang, Y. Xi, D. Wu, N. Deng, S. Yu, *et al.*, "Investigation of statistical retention of filamentary analog rram for neuromophic computing," in *2017 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2017, pp. 39–4.

[105] Y. Y. Chen, M. Komura, R. Degraeve, B. Govoreanu, L. Goux, A. Fantini, N. Raghavan, S. Clima, L. Zhang, A. Belmonte, *et al.*, "Improvement of data retention in HfO_2/Hf 1T1R RRAM cell under low operating current," in *2013 IEEE International Electron Devices Meeting*, IEEE, 2013, pp. 10–1.

[106] A. Prakash, D. Jana, and S. Maikap, "TaO_x-based resistive switching memories: Prospective and challenges," *Nanoscale research letters*, vol. 8, no. 1, p. 418, 2013.

[107] R. A. Cobley, C. D. Wright, and J. A. V. Diosdado, "A model for multilevel phase-change memories incorporating resistance drift effects," *IEEE Journal of the Electron Devices Society*, vol. 3, no. 1, pp. 15–23, 2014.

[108] I. Giannopoulos, A. Sebastian, M. Le Gallo, V. Jonnalagadda, M. Sousa, M. Boon, and E. Eleftheriou, "8-bit precision in-memory multiplication with projected phase-change memory," in *2018 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2018, pp. 27–7.

[109] S. Ambrogio, M. Gallot, K. Spoon, H. Tsai, C. Mackin, M. Wesson, S. Kariyappa, P. Narayanan, C.-C. Liu, A. Kumar, *et al.*, "Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks," in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2019, pp. 6–1.

[110] S. Choi, S. H. Tan, Z. Li, Y. Kim, C. Choi, P.-Y. Chen, H. Yeon, S. Yu, and J. Kim, "SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations," *Nature materials*, vol. 17, no. 4, pp. 335–340, 2018.

[111] K. Ni, M. Jerry, J. A. Smith, and S. Datta, "A circuit compatible accurate compact model for ferroelectric-FETs," in *2018 IEEE Symposium on VLSI Technology*, IEEE, 2018, pp. 131–132.